RANCANG BANGUN SISTEM PENCARIAN KOLEKSI LAPORAN SKRIPSI DAN PKL DENGAN TEKNOLOGI WEB SEMANTIK (STUDI KASUS: RUANG BACA FAKULTAS ILMU KOMPUTER UNIVERSITAS BRAWIJAYA)

SKRIPSI

Disusun oleh:
Prasetyo Iman Nugroho
NIM: 135150200111119

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

RANCANG BANGUN SISTEM PENCARIAN KOLEKSI LAPORAN SKRIPSI DAN PKL DENGAN TEKNOLOGI WEB SEMANTIK (STUDI KASUS: RUANG BACA FAKULTAS ILMU KOMPUTER UNIVERSITAS BRAWIJAYA)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Prasetyo Iman Nugroho
NIM: 135150200111119

Skripsi ini telah diuji dan dinyatakan lulus pada
11 Januari 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Bayu Priyambadha, S.Kom, M.Kom	Nanang Yudi Setiawan, S.T, M.Kom
NIP. 19820909 200812 1 004	NIP. 19760619 200604 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP. 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 23 November 2017

Prasetyo Iman Nugroho

NIM: 135150200111119

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Allah SWT yang telah melimpahkan kasih dan saying-Nya sehingga penulis dapat menyelesaikan penelitian untuk memenuhi skripsi dengan judul "RANCANG BANGUN SISTEM PENCARIAN KOLEKSI LAPORAN SKRIPSI DAN PKL DENGAN TEKNOLOGI WEB SEMANTIK (STUDI KASUS: RUANG BACA FAKULTAS ILMU KOMPUTER UNIVERSITAS BRAWIJAYA)". Penulis ingin menyampaikan terima kasih kepada pihak-pihak yang telah memberi bantuan dan dukungan atas selesainya penelitian ini, diantaranya:

1. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D selaku Dekan Fakultas Ilmu Komputer yang telah membantu selama menjalani perkuliahan di Fakultas Ilmu Komputer Universitas Brawijaya serta dalam pengesahan skripsi.
2. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika yang membantu selama menjalani perkuliahan di Fakultas Ilmu Komputer Universitas Brawijaya serta dalam pengesahan skripsi.
3. Bapak Bayu Priyambadha, S.Kom, M.Kom dan Bapak Nanang Yudi Setiawan, S.T, M.Kom selaku dosen pembimbing skripsi yang telah memberikan arahan dan bimbingan kepada penulis dalam mengerjakan penelitian ini.
4. Bapak Aryo Pinandito, S.T, M.MT selaku dosen penasihat akademik yang telah memberikan nasihat, saran, dan masukan dalam menjalani perkuliahan di Fakultas Ilmu Komputer Universitas Brawijaya.
5. Bapak Andi Susatyo dan Ibu Eka Satria Puspitarini selaku orang tua penulis yang telah memberikan doa, motivasi, serta dukungan moril dan materiil dalam pengerjaan penelitian ini.
6. Adik serta keluarga dan kerabat yang senantiasa memberikan doa serta dukungan semangat kepada penulis.
7. Segenap dosen program studi Teknik Informatika yang telah mengajarkan ilmu kepada penulis selama menjalani perkuliahan, serta staff dan civitas akademika yang telah membantu dalam proses kegiatan perkuliahan.
8. Seluruh teman-teman Teknik Informatika angkatan 2013 atas bantuan, dukungan, dan saran selama perkuliahan dan pengerjaan skripsi.
9. Seluruh teman-teman dari Ini Grup Positip dan Selflie atas dukungan dan saran selama perkuliahan dan pengerjaan skripsi.
10. Semua pihak lain yang telah banyak membantu dalam penyusunan skripsi ini.

Malang, 23 November 2017

Penulis

pras_iman@hotmail.com

ABSTRAK

Ruang Baca FILKOM Universitas Brawijaya mempunyai banyak koleksi yang dapat berupa laporan skripsi dan PKL dengan judul dan topik yang beragam. Saat ini sudah terdapat sistem pencarian yang digunakan untuk mencari koleksi yang ada di Ruang Baca FILKOM. Cara kerja dari sistem ini adalah pengguna memasukkan kata kunci (keywords), kemudian sistem akan melakukan pencarian berdasarkan keywords tadi lalu menampilkan informasi hasil pencarian. Namun hasil pencarian yang didapatkan bisa saja tidak relevan dari kata kunci tersebut dan tidak sesuai dengan persepsi atau ekspektasi dari pengguna. Dari permasalahan tersebut terdapat solusi yaitu pengembangan pengetahuan dari informasi koleksi secara semantik yang kemudian diakses dengan menggunakan sistem berbasis web. Ontologi dibangun berdasarkan data koleksi dengan menggunakan Protégé dan sistem dibuat dengan menggunakan PHP. Hasil pengujian white box dengan pengujian unit didapatkan hasil kompleksitas logika yang rendah. Sedangkan pada pengujian black box dengan pengujian validasi didapatkan hasil seluruh kasus uji adalah valid.

Kata kunci: web semantik, sistem pencarian, ontologi, Protégé

ABSTRACT

Library in Faculty of Computer Science (FILKOM) Universitas Brawijaya has many collections which may consists of thesis documents or internship reports with various titles and subjects. Currently there is a search system used to search collections which housed in the library. The system will receive keywords from user and then will perform a search based on the keyword and display the search results. However, the obtained search results may not be relevant from the keyword and not in accordance with user's perceptions and expectations. From this problem there is a solution which is to develop knowledge from library collections with semantic which will be accessed with a web-based system. The ontology is build on collection data using Protégé and the system developed using PHP. Result from white box test method using unit test showed that the system has low logic complexity. Meanwhile with black box test method on validation test, the results from all test cases are valid.

Keywords: semantic web, search system, ontology, Protégé

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE PROGRAM

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Web adalah suatu ruang informasi yang dimana terdapat sumber daya yang berguna yang saat ini sudah menjadi suatu kebutuhan bagi masyarakat baik untuk melakukan pencarian informasi, penyebaran informasi, hingga transaksi. Dalam pencarian informasi, web sudah dapat diimplementasikan dalam berbagai aspek dengan skala tertentu, dimulai dari skala besar seperti *Google* sampai skala kecil seperti sistem pencarian koleksi perpustakaan, contohnya di Ruang Baca Fakultas Ilmu Komputer (FILKOM) Universitas Brawijaya.

Ruang Baca FILKOM Universitas Brawijaya mempunyai banyak koleksi yang dapat berupa laporan skripsi dan PKL dengan judul dan topik yang beragam. Saat ini sudah terdapat sistem pencarian yang digunakan untuk mencari koleksi yang ada di Ruang Baca FILKOM. Cara kerja dari sistem ini adalah pengguna memasukkan kata kunci atau *keyword*, kemudian sistem akan melakukan pencarian berdasarkan *keyword* tadi lalu menampilkan informasi hasil pencarian. Namun hasil pencarian yang didapatkan bisa saja tidak relevan dari kata kunci tersebut dan tidak sesuai dengan persepsi atau ekspektasi dari pengguna.

Dari permasalahan diatas terdapat solusi yaitu pengembangan pengetahuan dari informasi koleksi secara semantik. Web Semantik merupakan suatu teknologi web yang dimana informasi diberi makna dan diberikan ketentuan secara logis sehingga mesin dapat mengerti dan mampu memproses informasi secara otomatis dan membuat mesin lebih mudah untuk mengintegrasi informasi yang tersedia (Maedche dan Staab 2005). Pada Web Semantik terdapat beberapa teknologi yang sudah ditetapkan sebagai standar W3C seperti RDF (*Resource Definition Framework*) untuk representasi data berupa *triple* (subjek-predikat-objek). Selain itu terdapat ontologi yang merupakan peranan penting dalam pembentukan basis pengetahuan yang kemudian direpresentasi dengan menggunakan RDFS (*Resource Definition Framework Schema*) dan OWL (*Web Ontology Language*). Semantik dapat memberikan peran yang dimana dari *keyword* yang dimasukkan oleh pengguna mampu didapat hasil yang relevan dan kontekstual dengan memahami makna dari *keyword* yang diberikan.

Penelitian sebelumnya mengenai Web Semantik telah dilakukan oleh Goldschmidt dan Krishnamoorthy (2005). Penelitian ini menghasilkan prototipe mesin pencari Web Semantik (*Semantic Web Search Engine* atau SWSE). Pada pengujiannya digunakan skenario uji yaitu penyelesaian teka-teki silang dengan menggunakan Google dan SWSE. Pada pencarian jawaban dengan menggunakan Google didapat kandidat jawaban untuk teka-teki silang mengenai presiden Amerika Serikat dengan rata-rata 107.69. Sedangkan pencarian jawaban dengan menggunakan SWSE untuk teka-teki yang sama didapat hasil kandidat yang lebih sedikit daripada Google dengan rata-rata 5.31. Pada tahun 2013, Arwan, et al dalam jurnalnya berjudul *Ontology and Semantic Matching for Diabetic Food Recommendations* mengenalkan sistem rekomendasi makanan untuk pengidap

diabetes. Sistem yang dihasilkan dari penelitian ini menggunakan teknologi Web Semantik sebagai basis domain pengetahuan dan metode Weighted Tree Similarity untuk akurasi pemahaman basis pengetahuan. Hasil dari penelitian ini adalah dari 30 sampel data pengidap diabetes, sistem mampu memberikan rekomendasi makanan kepada 73% dari sampel data (Arwan, et al. 2013).

Berdasarkan penelitian yang sudah dijelaskan menunjukkan bahwa Web Semantik dapat digunakan untuk pengembangan sistem pencarian koleksi. Untuk pengujiannya akan dilakukan perbandingan hasil pencarian berbasis kata kunci dan hasil pencarian berbasis semantik untuk mengetahui akurasi dari masing-masing metode. Harapannya, hasil dari penelitian ini adalah kemudian dapat membantu pengguna dalam mencari laporan skripsi dan PKL agar didapat hasil pencarian yang sesuai dengan kebutuhan dan juga dapat dijadikan sebagai referensi dasar bagi peneliti dalam penelitian selanjutnya mengenai Web Semantik. Penulis mempunyai gagasan untuk memberikan judul penelitian ini menjadi "**Rancang Bangun Sistem Pencarian Koleksi Laporan Skripsi dan PKL Menggunakan Teknologi Web Semantik**".

1.2 Rumusan Masalah

Berdasarkan permasalahan yang telah dijelaskan pada latar belakang, maka didapatkan rumusan masalah sebagai berikut:

1. Bagaimana analisis dan rancangan sistem pencarian koleksi laporan skripsi dan PKL dengan teknologi Web Semantik?
2. Bagaimana implementasi sistem pencarian koleksi laporan skripsi dan PKL dengan teknologi Web Semantik?
3. Bagaimana menguji sistem pencarian koleksi laporan skripsi dan PKL dengan teknologi Web Semantik?

1.3 Tujuan

Tujuan umum dari penelitian ini adalah untuk mengembangkan sistem pencarian koleksi laporan skripsi dan PKL dengan teknologi Web Semantik.

Sedangkan tujuan khusus dari penelitian ini adalah sebagai berikut:

1. Mengetahui analisis kebutuhan untuk sistem pencarian koleksi laporan skripsi dan PKL dengan teknologi Web Semantik.
2. Merancang dan mengimplementasikan sistem pencarian koleksi laporan skripsi dan PKL dengan teknologi Web Semantik.
3. Menguji sistem pencarian koleksi laporan skripsi dan PKL dengan teknologi Web Semantik.

1.4 Manfaat

Manfaat yang dapat diperoleh dari penelitian ini adalah sebagai berikut:

1. Bagi pengguna khususnya pengunjung Ruang Baca Fakultas Ilmu Komputer Unversitas Brawijaya dalam mencari koleksi laporan skripsi dan PKL agar didapat hasil pencarian yang sesuai dengan kebutuhan.
2. Hasil dari penelitian ini dapat dijadikan sebagai referensi dasar bagi peneliti dalam penelitian selanjutnya mengenai Web Semantik.

1.5 Batasan Masalah

Agar permasalahan yang dirumuskan dapat lebih terfokus, maka dalam penelitian ini dilakukan pembatasan pada hal-hal sebagai berikut:

1. Sistem yang akan dibangun merupakan sistem berbasis web.
2. Sistem hanya memberikan informasi terkait suatu koleksi yaitu judul, NIM penulis, tahun, dan nomor panggil.

1.6 Sistematika Pembahasan

Penyusunan skripsi ini menggunakan kerangka pembahasan yang disusun sebagai berikut:

Bab I **Pendahuluan**

Menjelaskan latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat, dan sistematika pembahasan dari penelitian.

Bab II **Landasan Kepustakaan**

Menjelaskan kajian pustaka, teori, dan referensi yang berhubungan dengan penelitian.

Bab III **Metodologi**

Menjelaskan alur metodologi yang dilakukan untuk menyelesaikan penelitian.

Bab IV **Analisis Kebutuhan**

Menjelaskan hal-hal yang terkait dengan analisis kebutuhan dari sistem.

Bab V **Perancangan**

Menjelaskan hal-hal yang terkait dengan perancangan dari sistem.

Bab VI **Implementasi**

Menjelaskan hal-hal yang terkait dengan implementasi sistem.

Bab VII **Pengujian**

Menjelaskan tentang teknik atau metode pengujian yang dilakukan pada penelitian.

Bab VIII **Penutup**

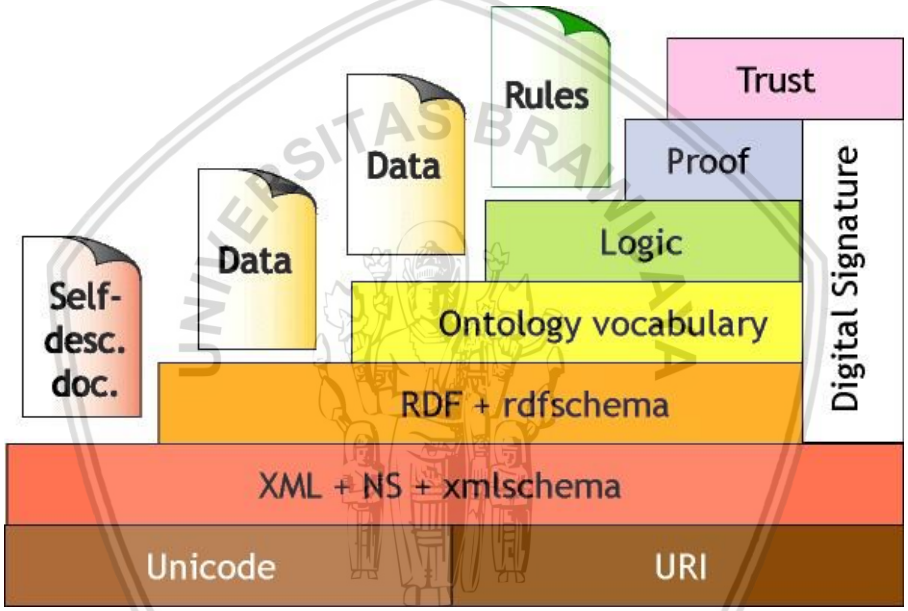
Menjelaskan kesimpulan yang dapat diambil dari penelitian skripsi disertai saran yang dapat dijadikan masukan untuk penelitian lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Web Semantik

Web Semantik adalah suatu teknologi yang dimana suatu informasi terhubung dengan informasi lainnya dengan suatu cara tertentu dan dimengerti oleh mesin, sehingga dapat diproses menjadi suatu basis pengetahuan. Web Semantik pertama kali dikenalkan oleh Tim Berners-Lee yang merupakan penemu dari *World Wide Web* (Antoniou dan Harmelen 2003).

Dalam Web Semantik terdapat struktur yang sudah ditetapkan oleh *W3C* (*World Wide Web Consortium*), yang dapat dilihat pada Gambar 2.1.



Gambar 2.1 Struktur Web Semantik

Sumber: (Antoniou dan Harmelen 2003)

Dari Gambar 2.1 dapat dijelaskan sebagai berikut (Antoniou dan Harmelen 2003):

- XML merupakan bahasa yang digunakan untuk membuat dokumen secara terstruktur dengan kamus yang didefinisikan oleh pengguna. XML cocok untuk pengiriman dokumen di Web;
- RDF menyediakan pemaknaan sederhana untuk model data (*resource*), dan dapat ditulis dengan sintaks XML;
- RDF Schema menyediakan kamus kosakata untuk RDF. Kunci utama dari RDF Schema adalah kelas dan properti, hubungan sub-kelas dan sub-properti, dan batasan *domain* dan *range*;
- *Ontology languages* memperluas fungsi dari RDF Schema dan mampu merepresentasikan hubungan antar *resource* lebih kompleks;

- *Logic* digunakan untuk menulis deklarasi pengetahuan dari *resource* dan dapat meningkatkan *ontology language*;
- *Proof* melibatkan proses deduktif nyata sama dengan representasi bukti pada bahasa web dan validasi bukti; dan
- *Trust* bertujuan untuk memastikan dan memverifikasi bahwa pernyataan berasal dari sumber yang terpercaya dan dapat dicapai dengan menggunakan *digital signature* dari pernyataan RDF.

2.1.1 XML (*Extensible Markup Language*)

XML (*Extensible Markup Language*) merupakan bahasa *meta* dengan mendefinisikan *markup* yang digunakan untuk membuat dokumen secara terstruktur dengan kamus yang didefinisikan oleh pengguna. XML mempunyai kerangka umum seperti HTML yang berdasarkan *tag* dan cocok untuk pengiriman dokumen di Web (Antoniou dan Harmelen 2003). Berbeda dengan HTML, pengguna dapat mendefinisikan *tag* sesuai dengan kebutuhan masing-masing.

Pada dokumen XML terdapat *prolog*, elemen, dan *epilog* yang bersifat opsional. Untuk bagian *epilog* tidak akan dibahas lebih lanjut pada bagian ini. Pada *prolog* berisi deklarasi XML dan referensi opsional ke dokumen eksternal. Contohnya ada pada Kode Program 2.1.

Kode Program 2.1 Contoh *Prolog* pada XML

```
<?xml version="1.0" encoding="UTF-16" ?>
```

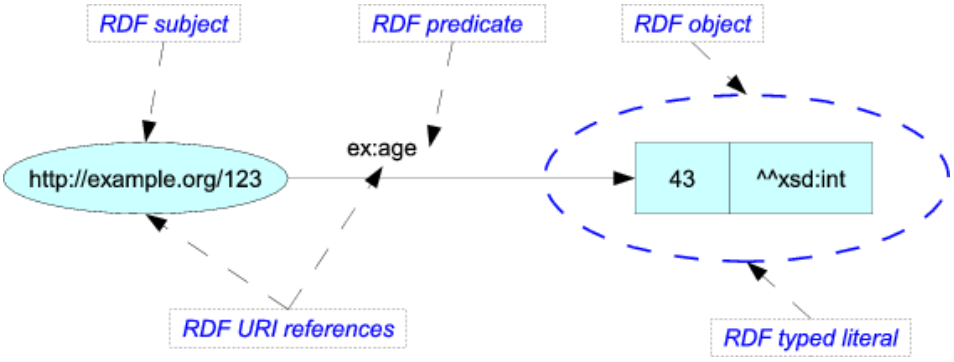
Bagian elemen terdiri dari *tag* pembuka, konten, dan *tag* penutup. Nama *tag* dapat dipilih secara bebas, namun dengan syarat harus diawali dengan huruf, garis bawah, atau tanda titik dua. Pada konten dapat berisi text, atau elemen lainnya, atau kosong. Contohnya ada pada Kode Program 2.2.

Kode Program 2.2 Contoh Konten pada XML

```
<lecturer>
  <name>David Billington</name>
  <phone> +61 - 7 - 3875 507 </phone>
</lecturer>
```

2.1.2 RDF (*Resource Definition Framework*)

RDF merupakan standar yang dibuat oleh W3C sebagai metode umum untuk memodelkan sebuah *resource* dengan sekumpulan format sintaks. RDF mewarisi sifat dari XML dalam membuat sintaks dan dapat mendefinisikan model data sederhana untuk representasi data yang dapat diproses oleh mesin (Fensel, et al. 2001). Konsep dasar dari RDF adalah bagaimana sebuah pernyataan (*statement*) dapat dibentuk dalam bentuk ekspresi subjek-predikat-objek atau biasa disebut dengan *triple*. Konsep dasar RDF dapat dilihat pada Gambar 2.2.



Gambar 2.2 Struktur *Statement* pada RDF

Sumber: (Nilsson, et al. 2008)

Dari Gambar 2.2 konsep dasar RDF dijelaskan lebih lanjut sebagai berikut:

- Subjek mengacu pada *resource* yang ingin dideskripsikan. Setiap *resource* mempunyai URI (*Universal Resource Identifier*) yang dapat berupa URL (*Universal Resource Locator*) atau alamat Web;
- Predikat mengacu pada *properties* yang menjelaskan relasi antar *resource* satu dengan lainnya. Predikat pada RDF juga diberikan URI untuk mengidentifikasi objek beserta hubungan-hubungannya; dan
- Objek mengacu pada nilai (*value*) yang menegaskan *property* dari sebuah *resource*. Nilai ini dapat berupa literal (*string* dan angka) atau *resource* lainnya.

RDF merupakan bahasa universal yang dapat digunakan untuk menjelaskan suatu *resource* menggunakan kosakata sesuai pengguna, namun tidak dapat berasumsi tentang suatu *resource* tertentu, begitu juga dalam melakukan pendefinisian makna dari banyak *resource*. Oleh karena itu, diperlukan kamus untuk mendefinisikan suatu *resource*. Kamus tersebut dibuat dengan suatu bahasa skema yaitu *Resource Definition Language Schema* (RDFS).

2.1.2.1 RDFS (*Resource Definition Language Schema*)

RDFS adalah bahasa ontologi primitif, yang dapat mendefinisikan suatu *resource* dengan makna pasti. Konsep utama dari RDFS adalah sebagai berikut:

1. *Classes*

   Dalam mendefinisikan sebuah *resource*, *class* digunakan untuk menjelaskan tipe suatu objek. Sebuah *class* dapat berisi banyak *resource*. Satu *resource* yang dimiliki oleh suatu *class* disebut *instance*. Suatu *class* dapat memiliki *subclass*, yang dimana *subclass* tersebut merupakan turunan (*inheritance*) dari *class* induk sehingga dapat membentuk sebuah hierarki *classes*.

2. *Properties*

   Property mendefinisikan relasi antar *resource*. Sebuah *property* dapat memiliki *subproperty* sama halnya dengan *class*. Pada RDFS, *properties*

didefinisikan secara global, yang dimana tidak terenkapsulasi sebagai atribut pada definisi suatu *class*. Jadi, sebuah *class* yang sudah dibuat dapat mendefinisikan *property* baru tanpa mengubah *class* tersebut.

Pada *properties* dapat diberikan batasan terhadap *resource* yang didefinisikan. *Domain* merupakan batasan untuk menjelaskan subjek *class* dari sebuah predikat *property* dalam *triple*. Sedangkan *range* merupakan batasan untuk menjelaskan objek *class* dari sebuah predikat *property* dalam *triple*.

Classes inti dari RDF dapat dilihat pada Tabel 2.1.

Tabel 2.1 *Classes* Inti dari RDF

Nama *class*	Penjelasan
rdfs:Resource	Merupakan *class* dari semua *resource*.
rdfs:Class	Merupakan *class* dari semua *classes*.
rdfs:Literal	Merupakan *class* dari semua literal (*string* dan angka).
rdf:Property	Merupakan *class* dari *properties* dalam RDF.
rdf:Statement	Merupakan *class* dari *statement* dalam RDF.

Sumber: (Antoniou dan Harmelen 2003)

Properties inti dari RDF dapat dilihat pada Tabel 2.2.

Tabel 2.2 *Properties* Inti dari RDF

Nama *class*	Penjelasan
rdf:type	Terkait dengan *resource* dari sebuah *class*
rdfs:subClassOf	Terkait dengan *class* turunan dari *class* induk (*superclass*).
rdfs:subPropertyOf	Terkait dengan *property* turunan dari *property* induk (*superproperty*).
rdfs:domain	Digunakan untuk mendefinisikan subjek *class* dari sebuah predikat *property* dalam *triple*.
rdfs:range	Digunakan untuk mendefinisikan objek *class* dari sebuah predikat *property* dalam *triple*.

Sumber: (Antoniou dan Harmelen 2003)

2.1.3 Ontologi

Ontologi merupakan konsep tentang makna dari suatu objek, properti dari suatu objek, serta relasi objek tersebut yang mungkin terjadi pada suatu domain

pengetahuan (Lanin dan Lyadova 2007). Hubungan antar objek menunjukkan bagaimana konsep ini berkorelasi satu sama lain dalam satu domain data. Ontologi memudahkan dalam berbagi pengetahuan dan penggunaan kembali yakni suatu pemahaman umum dari berbagai jenis konten yang menjembatani aplikasi dengan manusia (Ghaleb, et al. 2006).

Menurut Ghaleb, et al (2006) secara teknis, ontologi adalah suatu potongan pengetahuan secara teks, yang diletakan di web agar agent dapat berkonsultasi ke ontologi ketika diperlukan, dan mempresentasikan menggunakan sintak dari suatu penyajian bahasa ontologi. Menurut Ibrahim (2007), ontologi sendiri dapat didefinisikan sebagai suatu cara untuk mendeskripsikan arti dan relasi dari suatu domain. Deskripsi tersebut berisi *class*, *property*, dan *instance*. Deskripsi ini dapat membantu sistem komputer dalam menggunakan istilah-istilah tersebut dengan cara yang lebih mudah (Awaludin 2010).

2.1.3.1 OWL (*Web Ontology Language*)

OWL (*Web Ontology Language*) adalah suatu bahasa yang dapat digunakan oleh aplikasi-aplikasi yang bukan sekedar menampilkan informasi tersebut pada manusia, melainkan juga yang perlu memproses isi informasi isi. *Ontology* sendiri dapat didefinisikan sebagai suatu cara untuk mendeskripsikan arti dan relasi dari istilah-istilah. Dengan menggunakan OWL dapat menambah kosakata tambahan disamping semantik formal yang telah dibuat sebelumnya menggunakan XML, RDF, dan RDFS. Hal ini sangat membantu penginterpretasian mesin yang lebih baik terhadap isi web. Untuk mendeskripsikan *properties* dan *classes*, OWL menambahkan *vocabulary* seperti:

- "*among others*";
- Relasi antar *classes* (contohnya "*disjointness*");
- Kardinalitas (contohnya "*exactly one*");
- Kesamaan (*equality*);
- Karakteristik *property* (contohnya "*symmetry*"); dan
- *Enumerated classes*.

OWL menyediakan tiga buah subbahasa yang dirancang untuk digunakan oleh para pengguna tertentu, yaitu:

- OWL Lite, digunakan oleh pengguna yang membutuhkan suatu hierarki pengklasifikasian dan berbagai *constraints* sederhana;
- OWL DL, digunakan oleh pengguna yang menginginkan tingkat ekpresi maksimal dan semua konklusi yang dihasilkan dapat dihitung dalam waktu yang terbatas (*finite*); dan

- OWL Full, digunakan oleh pengguna yang menginginkan tingkat ekpresi maksimal dan kebebasan sintaks dari RDF tanpa mempertimbangkan komputasi yang dibutuhkan.

2.1.4 SPARQL

SPARQL Protocol and RDF Query Language (SPARQL) adalah sebuah protokol dan bahasa *query* untuk merujuk sumber daya dari aplikasi web semantik. Sebuah *query* yang menggunakan SPARQL dapat terdiri atas *triple patterns*, konjungsi (*or*), dan disjungsi (*and*).

Hasil dari query SPARQL dapat mengembalikan nilai dalam beberapa format data, antara lain XML, JSON, RDF, dan HTML (Awaludin 2010). Untuk menjalankan SPARQL, dapat menggunakan beberapa *tools* dan API seperti ARQ, Rasqal, RDF::Query, twingql, Pellet, dan KAON2 (Ibrahim 2007). *Tools* tersebut memiliki API yang memudahkan pengguna untuk memanipulasi hasil *query* dengan berbagai aplikasi yang ada. Namun, sebagai standar dapat menggunakan *SPARQL Query Results XML Format* (Ibrahim 2007) yang direkomendasikan oleh W3C.

Contoh dari *query* SPARQL untuk mencari ibukota dari tiap provinsi di Indonesia ditunjukkan pada Gambar 2.3.

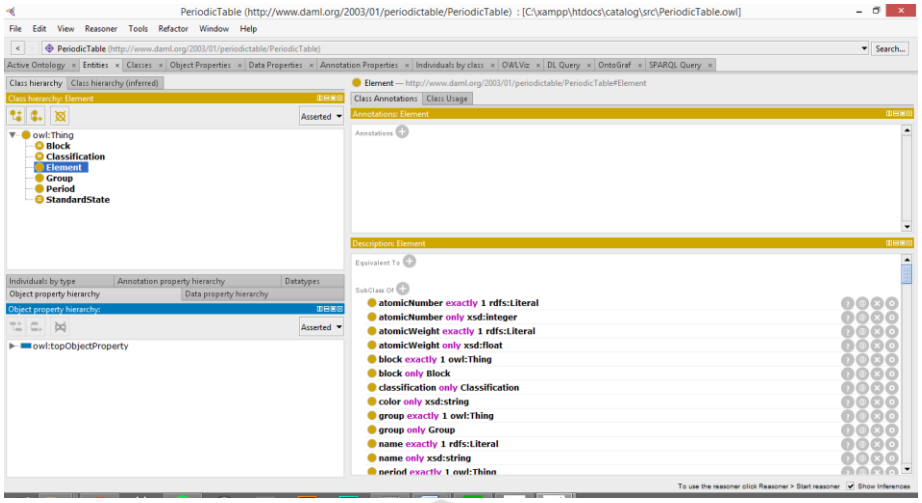
Gambar 2.3 Contoh *Query* SPARQL

Sumber: (Ibrahim 2007)

```
PREFIX abc: <http://mynamespace.com/exampleOntologie#>
SELECT ?capital ?province
WHERE {
  ?x abc:cityname ?capital.
  ?y abc:provincename ?province.
  ?x abc:isCapitalOf ?y.
  ?y abc:isInCountry abc:indonesia.
}
```

2.1.5 Protégé

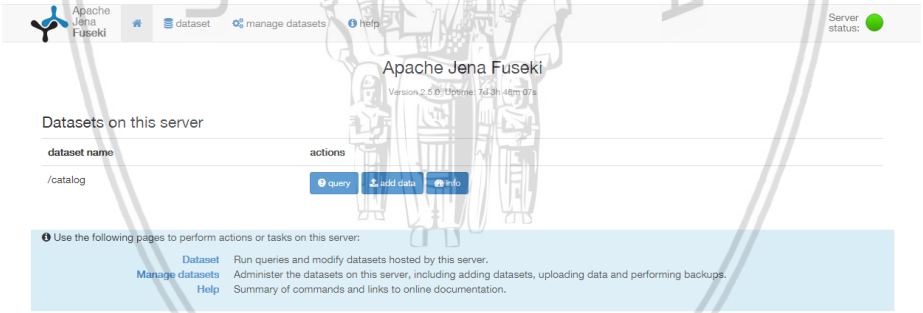
Protégé merupakan sebuah *tool* yang digunakan untuk membuat domain ontologi. Selain itu Protégé juga dapat melakukan *query* dengan menggunakan SPARQL. Protégé memiliki format penyimpanan seperti OWL, RDF, XML, Turtle, Manchester OWL, JSON-LD, LaTeX, dan OBO. Protégé dibuat dengan menggunakan bahasa pemrograman Java. Fungsi dalam Protégé dapat digunakan melalui *Graphical User Interface* (GUI) dengan menampilkan *tab* untuk masing-masing bagian dan fungsi standar.



Gambar 2.4 Protégé

2.1.6 Apache Jena Fuseki

Apache Jena Fuseki adalah *graph store server* yang menyediakan protokol SPARQL 1.1 untuk melakukan *query* dan *update* menggunakan protokol SPARQL 1.1 Graph Store HTTP. Apache Jena Fuseki dapat dijalankan sebagai *service* pada sistem operasi, Java *web application* (WAR), atau sebagai *standalone server*. Cara Apache Jena Fuseki dapat digunakan untuk menyediakan mesin protokol untuk sistem *query* dan penyimpanan RDF lainnya (Fuseki 2017).



Gambar 2.5 Apache Jena Fuseki

2.2 *Software Development Life Cycle* (SDLC)

Software development life cycle (SDLC) adalah istilah untuk menjelaskan tahapan-tahapan dalam proses pengembangan perangkat lunak. SDLC memiliki sebuah standar internasional yaitu ISO/IEC 12207. Standar ini bertujuan untuk mendefinisikan semua tugas yang diperlukan di dalam mengembangkan dan merawat sebuah perangkat lunak (Pressman 2001).

Adapun tahapan-tahapan dalam melakukan pengembangan perangkat lunak adalah sebagai berikut:

a. Perencanaan dan analisis kebutuhan, tahapan ini dilakukan dengan dasar informasi masukan dari *stakeholder* yang kemudian digunakan untuk perencanaan pengembangan proyek;

b. Mendefinisikan kebutuhan, tahapan yang dimana informasi dari analisis kebutuhan akan di desain dan dikembangkan;

c. Desain aritektur produk, tahapan ini dilakukan untuk mendesain bagaimana sebuah produk akan dibuat. Pada tahapan ini *stakeholder* akan mengulas apakah mereka ingin melakukan pembuatan proyek tersebut;

d. Membuat produk, tahapan dimana sebuah produk mulai dibuat. Pengembang akan membuat produk dengan mengikuti desain dan arsitektur yang sebelumnya telah disetujui;

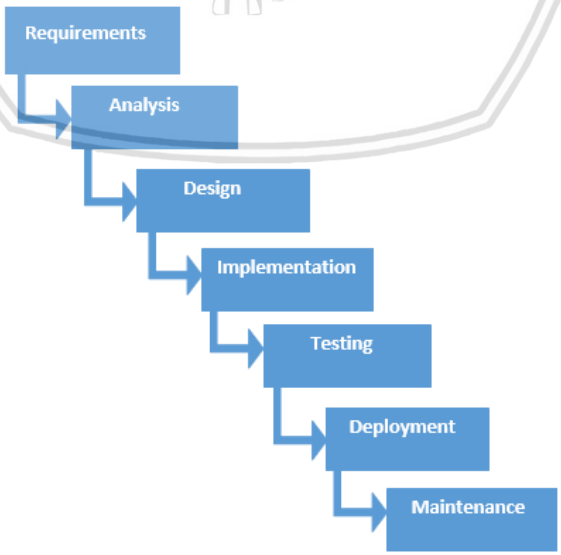
e. Pengujian produk, tahapan dimana produk akan diuji kelayakannya dan diukur seberapa baik produk tersebut dibuat; dan

f. Perawatan produk, tahapan dimana produk akan terus dirawat dan dikembangkan lagi.

2.2.1 *Waterfall Model*

Model *waterfall* dirancang oleh Winston W. Royce pada tahun 1970. Dalam model ini seluruh pekerjaan dilakukan dalam mode linear. Seluruh pekerjaan dibagi menjadi lima fase yang berbeda. Dalam model ini, tiap tahap harus dilakukan sebelum lanjut ke tahap selanjutnya. Semua tahapan mengalir dan berhubungan satu dengan lainnya, sebagai contoh suatu tahap akan dimulai setelah tahap sebelumnya telah selesai dilakukan.

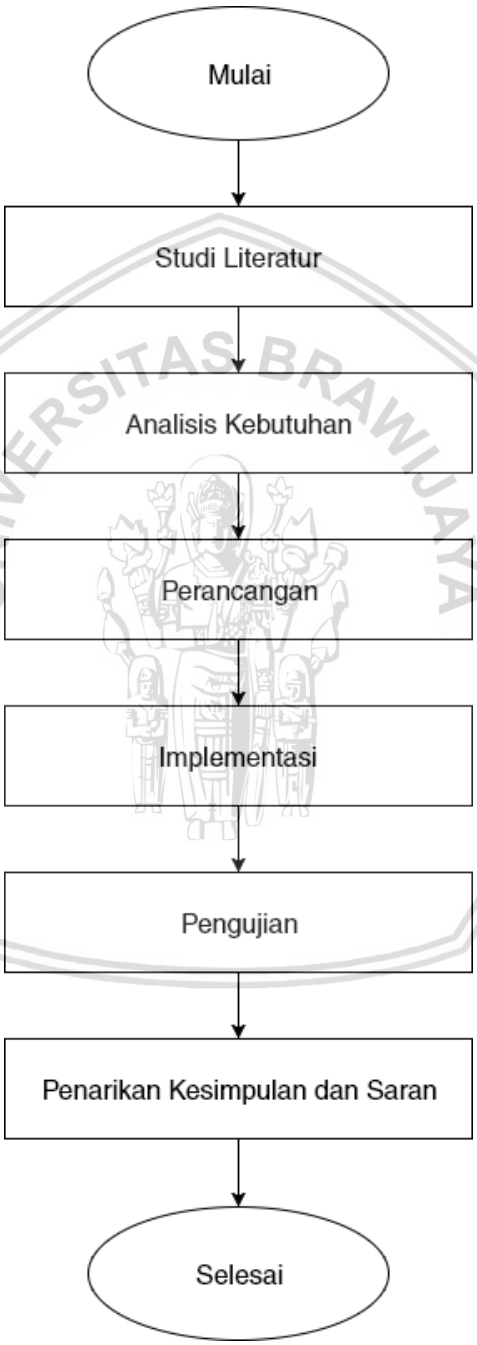
Model *waterfall* mempunyai tahapan pengembangan yang terstruktur, bertahap antara sub pengembangan pertama dengan berikutnya yang meliputi analisa sistem yang dilanjutkan dengan perancangan yang meliputi perancangan basisdata, perancangan proses, dan perancangan prosedur kerja, implementasi hasil perancangan, pengujian, dan pemeliharaan.



Gambar 2.6 *Waterfall Model*

BAB 3 METODOLOGI

Pada bab ini akan dijelaskan tentang langkah-langkah yang digunakan dalam penelitian ini. Metodologi penelitian ini terdiri dari enam tahap. Runtutan pengerjaan dari penelitian ini dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram Alur Metodologi Penelitian

3.1 Studi Literatur

Dalam tahap ini dilakukan kajian literatur dari beberapa topik yang digunakan sebagai acuan dalam penelitian ini seperti Web Semantik. Literatur yang menunjang tentang topik tersebut didapatkan dari berbagai sumber seperti buku, jurnal, dan artikel-artikel di internet yang memiliki relevansi dengan penelitian ini.

3.2 Analisis Kebutuhan

Analisa kebutuhan dilakukan untuk mengkaji kebutuhan apa saja yang dibutuhkan dari sistem yang akan dikembangkan. Proses ini dilakukan dengan menganalisis kebutuhan sistem pencarian koleksi laporan skripsi dan PKL. Analisis kebutuhan yang dilakukan ialah terkait dengan pengguna (aktor) dan kebutuhan fungsional dari sistem yang akan dikembangkan. Dari kebutuhan fungsional yang sudah dirumuskan kemudian dimodelkan ke dalam bentuk *use case diagram* dan *use case scenario*.

3.3 Perancangan

Perancangan perangkat lunak dilakukan setelah kebutuhan dari perangkat lunak didapatkan melalui tahap analisis kebutuhan. Pada tahapan ini dilakukan pemodelan *sequence diagram*, pemodelan *class diagram*, perancangan data, perancangan komponen, dan perancangan antarmuka. *Class* diagram digunakan untuk menggambarkan struktur *class* pada sistem, sedangkan *sequence diagram* digunakan untuk menggambarkan alur komunikasi antar *class* yang sudah dirancang. Pada perancangan data dilakukan perancangan basis pengetahuan (*knowledge base*) pada ontologi sebelum diimplementasikan ke dalam bentuk OWL (*ontology language*). Pada perancangan komponen dijelaskan atribut dan algoritma fungsi yang terdapat pada suatu *class* yang telah dimodelkan pada *class diagram*. Rancangan kasar dari tampilan halaman sistem yang dibangun juga akan dibahas pada bagian perancangan antarmuka.

3.4 Implementasi

Pada tahap ini akan dijelaskan proses implementasi sistem yang mengacu dari perancangan yang sudah dilakukan. Pada tahap ini dilakukan implementasi *class*, implementasi komponen yaitu fungsi-fungsi utama, implementasi ontologi, dan implementasi antarmuka. Sistem akan diimplementasikan dengan menggunakan bahasa pemrograman PHP. Implementasi *class* menjelaskan implementasi dari masing-masing *class* yang telah dirancang pada *class diagram*. Pada implementasi komponen, beberapa algoritme fungsi yang telah dirancang pada perancangan komponen akan diimplementasi. Implementasi ontologi dilakukan berdasarkan perancangan data yang telah dibuat. Kemudian pada implementasi antarmuka, rancangan yang sudah dibuat pada tahap perancangan antarmuka dibuat.

3.5 Pengujian

Pengujian dilakukan untuk mengukur kinerja dan performa dari perangkat lunak agar sesuai dengan kebutuhan yang telah didapatkan dan dengan implementasi yang telah dibuat. Strategi yang dilakukan dalam tahapan ini adalah dilakukan tiga jenis pengujian, yaitu pengujian unit, pengujian validasi, dan pengujian perbandingan sintaksis dan semantik. Pengujian unit dilakukan dengan metode *white box* dengan serangkaian kasus uji, sedangkan pengujian validasi dilakukan dengan menggunakan metode *black box*. Pada pengujian perbandingan sintaksis dan semantic dilakukan perbandingan hasil pencarian dengan kata kunci dan hasil pencarian dengan pendekatan semantik untuk mengetahui akurasi dari masing-masing metode. Hasil dari pengujian tersebut kemudian dianalisis untuk mendapatkan informasi serta kesimpulan atas pengujian yang telah dilakukan.

3.6 Penarikan Kesimpulan dan Saran

Dari perancangan, implementasi, serta pengujian terhadap sistem ini akan dilakukan penarikan kesimpulan terhadap hasil yang didapatkan selama proses penelitian. Kesimpulan berupa jawaban atas rumusan masalah yang telah ditentukan sebelumnya. Kemudian dilanjutkan dengan saran yang didapat dari kesalahan-kesalahan yang terjadi selama penelitian ini dilakukan agar dapat menyempurnakan penelitian selanjutnya jika pengembangan sistem selanjutnya akan dilakukan.

BAB 4 ANALISIS KEBUTUHAN

4.1 Gambaran Umum Sistem

Sistem yang akan dikembangkan adalah sistem pencarian koleksi laporan skripsi dan PKL. Sistem ini dirancang untuk membantu pengunjung Ruang Baca FILKOM untuk mencari koleksi laporan skripsi dan PKL yang ada disana. Pada sistem ini terdapat dua *stakeholder* yang terlibat, yaitu pengunjung dan admin. Pengunjung merupakan aktor yang dapat dapat melakukan pencarian informasi koleksi dengan menggunakan kata kunci (*keyword*). Sedangkan admin merupakan aktor yang berperan sebagai pengelola data koleksi.

4.2 Identifikasi Aktor

Pada tahap ini dijelaskan mengenai aktor-aktor yang berinteraksi dan menggunakan sistem ini. Daftar aktor beserta masing-masing deskripsinya dijelaskan pada Tabel 4.1.

Tabel 4.1 Identifikasi Aktor

Aktor	Deskripsi
Pengunjung	Merupakan aktor yang dapat menggunakan sistem untuk mencari informasi koleksi.
Admin	Merupakan aktor yang dapat menggunakan sistem untuk mengelola koleksi.

4.3 Kebutuhan Fungsional

Tahap ini dilakukan untuk mengetahui kebutuhan apa saja yang harus dipenuhi oleh sistem. Pada kebutuhan fungsional diberikan kode kebutuhan dengan format penulisan RBF-F-XXX, dengan XXX merupakan nomor dari kebutuhan yang ada. Daftar kebutuhan kebutuhan fungsional sistem dimuat pada Tabel 4.2.

Tabel 4.2 Kebutuhan Fungsional Sistem

Kode Fungsi	Nama Fungsi	*Stakeholder*	Deskripsi
RBF-F-001	Mencari informasi koleksi	Pengunjung	Sistem dapat mencari informasi koleksi dengan menggunakan kata kunci (*keyword*) sebagai parameter masukan dari pengguna. Informasi yang disediakan adalah judul koleksi, NIM penulis, jenis koleksi, nomor panggil

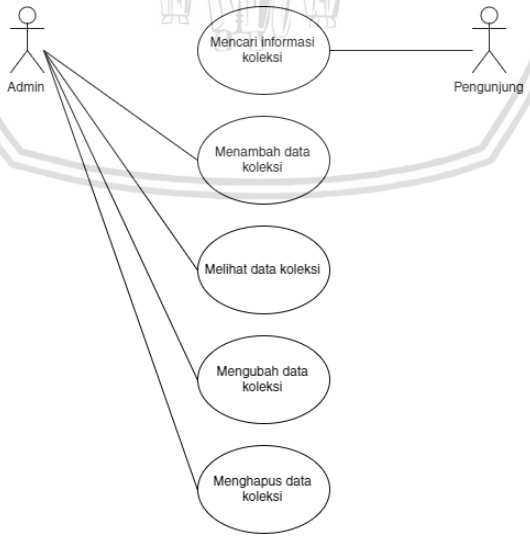
			koleksi, dan kata kunci terkait koleksi tersebut.
RBF-F-002	Menambah data koleksi	Admin	Sistem dapat menambah data koleksi laporan baru dan menyimpan dalam bentuk *triple*.
RBF-F-003	Melihat data koleksi	Admin	Sistem dapat menampilkan data koleksi laporan.
RBF-F-004	Mengubah data koleksi	Admin	Sistem dapat mengubah data koleksi laporan.
RBF-F-005	Menghapus data koleksi	Admin	Sistem dapat menghapus data koleksi laporan.

4.4 Pemodelan Kebutuhan

Untuk memudahkan pemahaman akan sistem yang akan dikembangkan, kebutuhan yang sudah disusun kemudian dimodelkan ke dalam bentuk diagram. Adapun diagram yang digunakan untuk memodelkan sistem adalah diagram *use case* dan *use case scenario*.

4.4.1 *Use Case Diagram*

Use case diagram berisi sejumlah aksi yang dapat dilakukan oleh pengguna dengan sistem dalam mencapai suatu kebutuhan tertentu. Diagram *use case* dari sistem ini ditunjukkan pada Gambar 4.1.



Gambar 4.1 *Use Case Diagram*

4.4.2 *Use Case Scenario*

Use case scenario merupakan penjabaran detil dari tiap *use case* yang ada di dalam sistem. Pada umumnya *use case scenario* terdiri dari nama *use case*, kode kebutuhan, aktor yang terlibat, tujuan, *pre-condition*, *main flow*, *post-condition*, dan *alternative flow*. *Pre-condition* merupakan kondisi awal yang harus dipenuhi sebelum aktor melakukan *use case*. *Alternative flow* merupakan alur alternatif dari *use case* jika kondisi tidak sesuai dengan *main flow*. *Post-condition* merupakan kondisi akhir setelah aktor melakukan *use case*. Skenario *use-case* yang ada pada sistem dijelaskan pada Tabel 4.3, Tabel 4.4, Tabel 4.5, Tabel 4.6, dan Tabel 4.7.

Tabel 4.3 *Use Case Scenario* Mencari Informasi Koleksi

Nama Use Case	Mencari informasi koleksi
Kode Kebutuhan	RBF-F-001
Aktor	Pengunjung
Tujuan	Mencari informasi koleksi sesuai yang diinginkan oleh aktor.
Pre-condition	-
Main flow	1. Aktor memasukkan *keyword* pada *search field*. 2. Aktor menekan tombol *Search*.
Post-condition	Sistem akan menampilkan hasil pencarian koleksi.
Alternative flow	1. Apabila aktor tidak mengisi *keyword* pada *search field* ketika menekan tombol *Search*, maka sistem akan mengirim notifikasi bahwa kolom *search field* kosong dan aktor tidak akan diarahkan ke halaman hasil pencarian.

Tabel 4.4 *Use Case Scenario* Menambah Data Koleksi

Nama Use Case	Menambah data koleksi
Kode Kebutuhan	RBF-F-002
Aktor	Admin
Tujuan	Menambah data koleksi baru.
Pre-condition	Aktor sudah berada di halaman daftar koleksi.
Main flow	1. Aktor menekan tombol *Add New* untuk menampilkan halaman *popup* form tambah koleksi baru. 2. Aktor mengisi data yang dibutuhkan pada form. 3. Aktor menekan tombol *Submit* setelah mengisi form.

Post-condition	Sistem akan menampilkan notifikasi bahwa koleksi baru berhasil ditambahkan dan sistem akan memperbarui halaman daftar koleksi.
Alternative flow	1. Apabila aktor tidak mengisi seluruh *field* yang dibutuhkan, ketika menekan tombol *Submit*, maka sistem akan menampilkan notifikasi bahwa *field* kosong dan sistem tidak akan melakukan apapun.

Tabel 4.5 *Use Case Scenario* Melihat Data Koleksi

Nama Use Case	Melihat data koleksi
Kode Kebutuhan	RBF-F-003
Aktor	Admin
Tujuan	Melihat data koleksi.
Pre-condition	Aktor sudah masuk ke dalam sistem sebagai admin.
Main flow	Aktor membuka halaman daftar koleksi.
Post-condition	Sistem akan menampilkan hasil pencarian koleksi.
Alternative flow	-

Tabel 4.6 *Use Case Scenario* Mengubah Data Koleksi

Nama Use Case	Mengubah data koleksi
Kode Kebutuhan	RBF-F-004
Aktor	Admin
Tujuan	Mengubah suatu data koleksi yang sudah ada.
Pre-condition	Aktor sudah berada di halaman daftar koleksi.
Main flow	1. Aktor menekan tombol *Edit* pada kolom *Action* untuk menampilkan tampilan *popup* form ubah data koleksi. 2. Aktor mengisi data yang akan diubah pada form. 3. Aktor menekan tombol *Submit* setelah mengisi form.
Post-condition	Sistem akan menampilkan notifikasi bahwa data koleksi berhasil diperbarui dan sistem akan memperbarui halaman daftar koleksi.
Alternative flow	Apabila aktor tidak mengisi seluruh *field* yang dibutuhkan, ketika menekan tombol *Submit*, maka sistem akan mengirimkan notifikasi bahwa ada *field* kosong dan sistem tidak akan melakukan apapun.

Tabel 4.7 *Use Case Scenario* Menghapus Data Koleksi

Nama Use Case	Menghapus data koleksi
Kode Kebutuhan	RBF-F-005
Aktor	Admin
Tujuan	Menghapus data koleksi yang sudah ada.
Pre-condition	Aktor sudah berada di halaman daftar koleksi.
Main flow	1. Aktor menekan tombol *Delete* pada kolom *Action* di dalam tabel koleksi untuk menampilkan tampilan *popover* berisi konfirmasi untuk menghapus data. 2. Aktor menekan tombol *Yes* pada tampilan *popover*.
Post-condition	Sistem akan menampilkan notifikasi bahwa data koleksi berhasil dihapus dan sistem akan memperbarui halaman daftar koleksi.
Alternative flow	Apabila aktor memilih *No* pada halaman *popover* konfirmasi maka sistem akan menutup halaman *popover* dan proses penghapusan data koleksi dibatalkan.

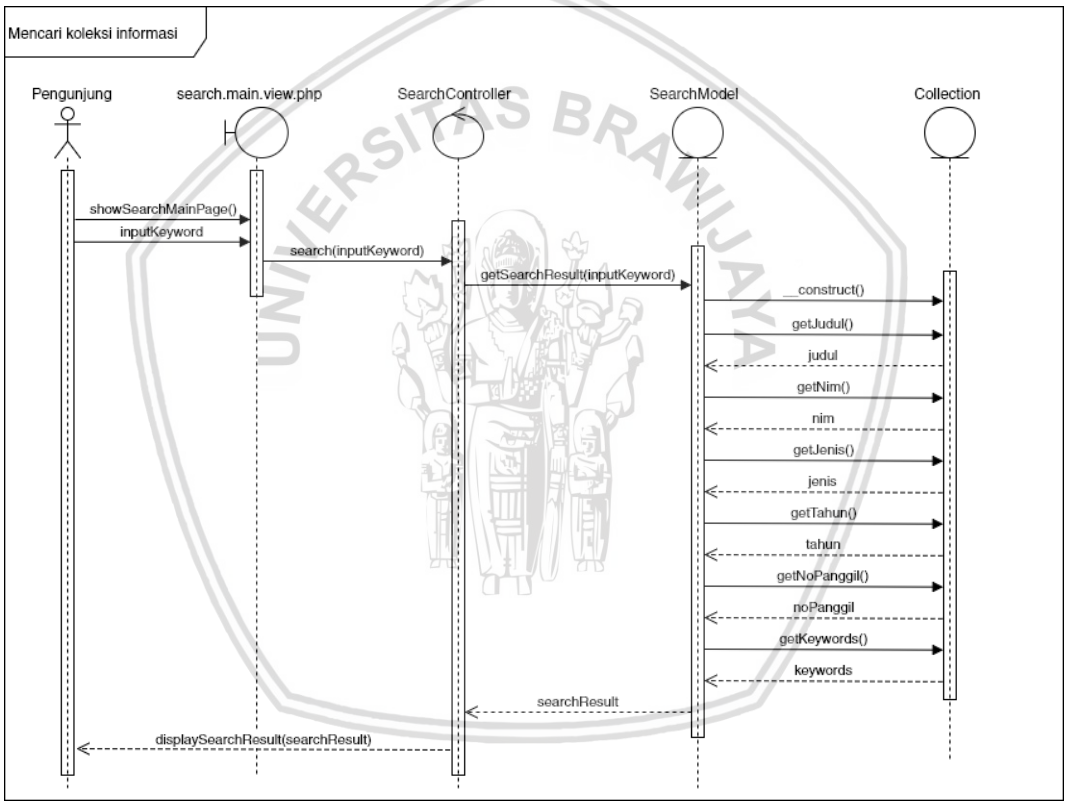
BAB 5 PERANCANGAN

5.1 UML (*Unified Modelling Language*)

Dari kebutuhan yang sudah dirancang, kebutuhan yang sudah disusun kemudian dimodelkan ke dalam bentuk diagram UML. Adapun diagram UML yang digunakan untuk memodelkan sistem adalah *sequence diagram* dan *class diagram*.

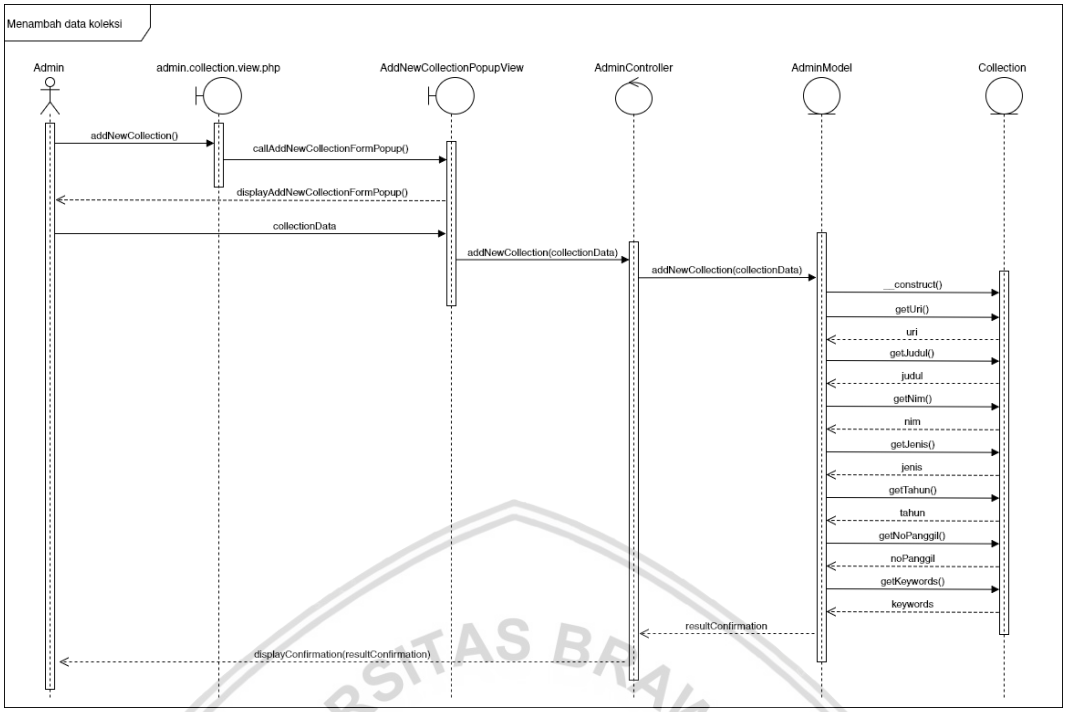
5.1.1 Sequence Diagram

Sequence diagram digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah dalam suatu *use case*. *Sequence diagram* sistem ini dijelaskan pada Gambar 5.1, Gambar 5.2, Gambar 5.3, Gambar 5.4, dan Gambar 5.5.



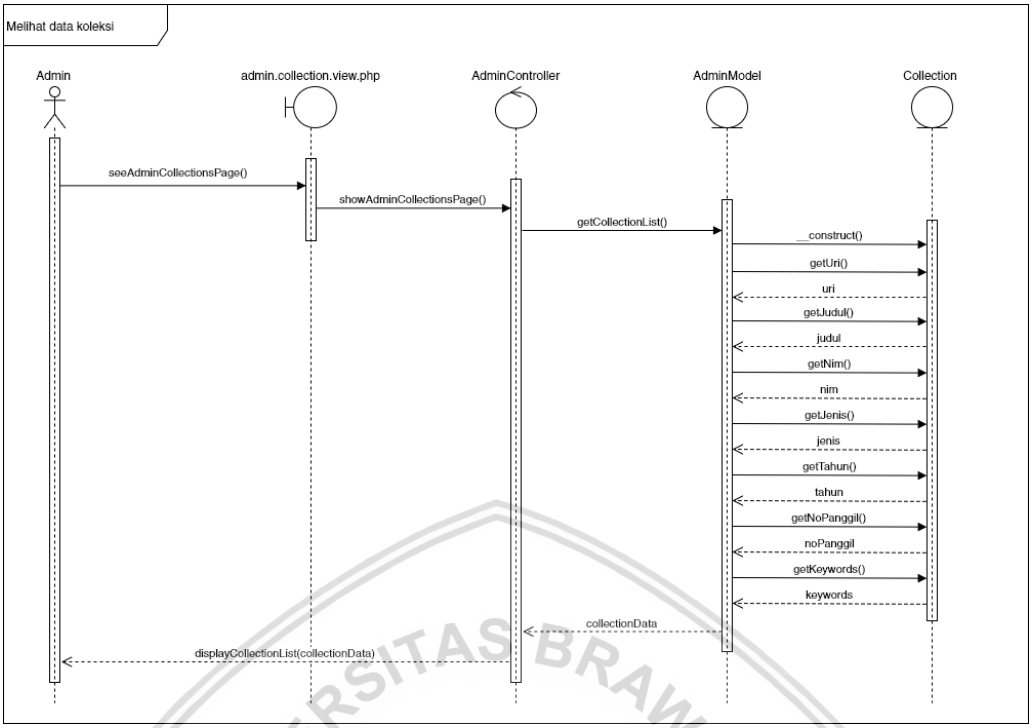
Gambar 5.1 *Sequence Diagram* Mencari Informasi Koleksi

Gambar 5.1 menggambarkan interaksi aktor (pengunjung) yang melakukan aktifitas pencarian informasi koleksi. Pertama aktor membuka halaman utama pencarian dan kemudian memasukkan *keyword*. Kelas SearchController kemudian melakukan proses untuk mendapatkan hasil pencarian dan mengarahkan aktor ke halaman hasil pencarian.



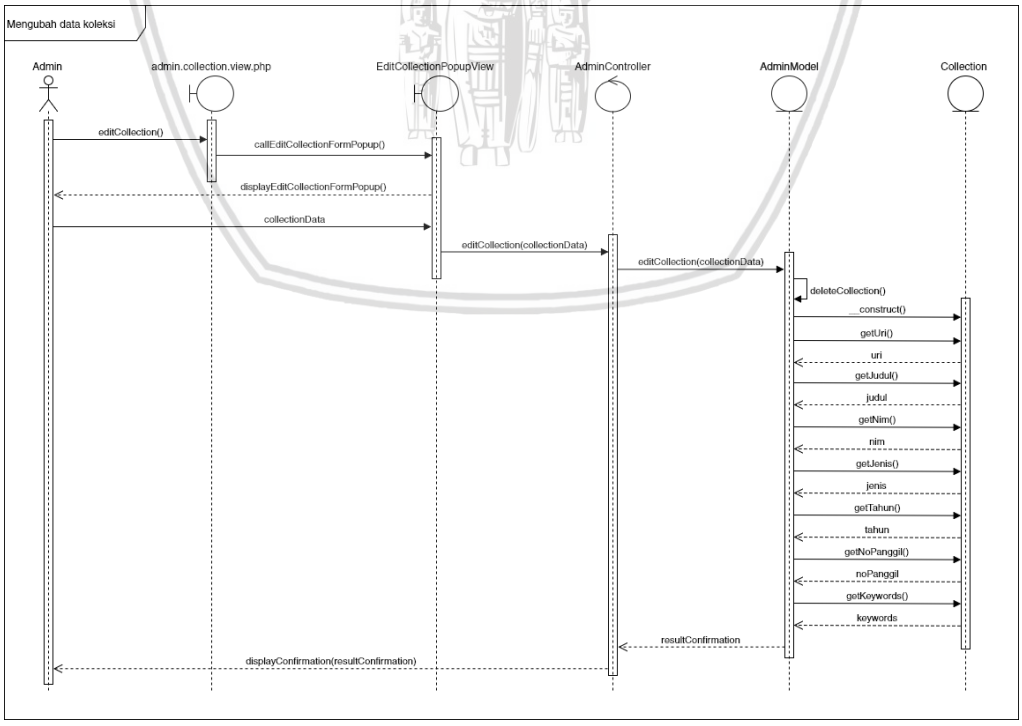
Gambar 5.2 *Sequence Diagram* Menambah Data Koleksi

Gambar 5.2 menggambarkan interaksi aktor (admin) yang melakukan aktifitas menambah data koleksi. Pertama aktor menekan tombol *Add New* di halaman daftar koleksi maka sistem akan menampilkan halaman *popup* tambah data koleksi. Lalu aktor mengisi informasi yang dibutuhkan dan menekan tombol *Submit*. Kelas AdminController akan meneruskan informasi ke kelas AdminModel untuk diperbarui. Pada kelas AdminModel informasi yang akan diperbarui terlebih dahulu sebelum diberikan informasi baru. Kemudian AdminModel akan mengembalikan konfirmasi hasil penyimpanan ke AdminController untuk kemudian ditampilkan.



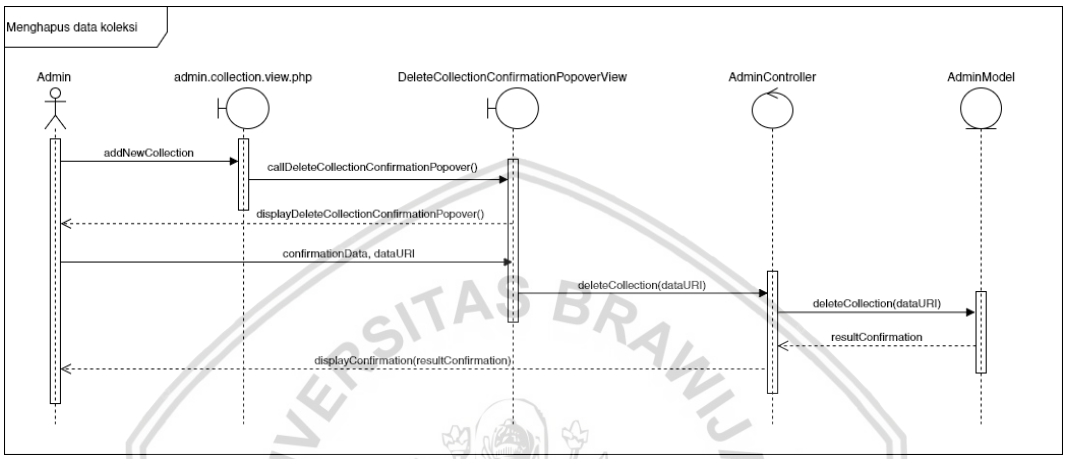
Gambar 5.3 *Sequence Diagram* Melihat Data Koleksi

Gambar 5.3 menggambarkan interaksi aktor (pengunjung) yang melakukan aktifitas melihat data koleksi. Pertama aktor membuka halaman daftar koleksi dan kelas AdminController akan meminta data koleksi ke AdminModel. Data kemudian di-*fetch* dan ditampilkan ke halaman daftar koleksi.



Gambar 5.4 *Sequence Diagram* Mengubah Data Koleksi

Gambar 5.4 menggambarkan interaksi aktor (pengunjung) yang melakukan aktifitas mengubah data koleksi. Pertama aktor menekan tombol *Edit* pada baris data koleksi di halaman daftar koleksi dan sistem akan menampilkan halaman *popup* ubah data koleksi. Lalu aktor menmperbarui informasi yang ada dan menekan tombol *Submit*. Kelas AdminController akan meneruskan informasi ke kelas AdminModel untuk disimpan. Kemudian AdminModel akan mengembalikan konfirmasi hasil penyimpanan ke AdminController untuk kemudian ditampilkan.

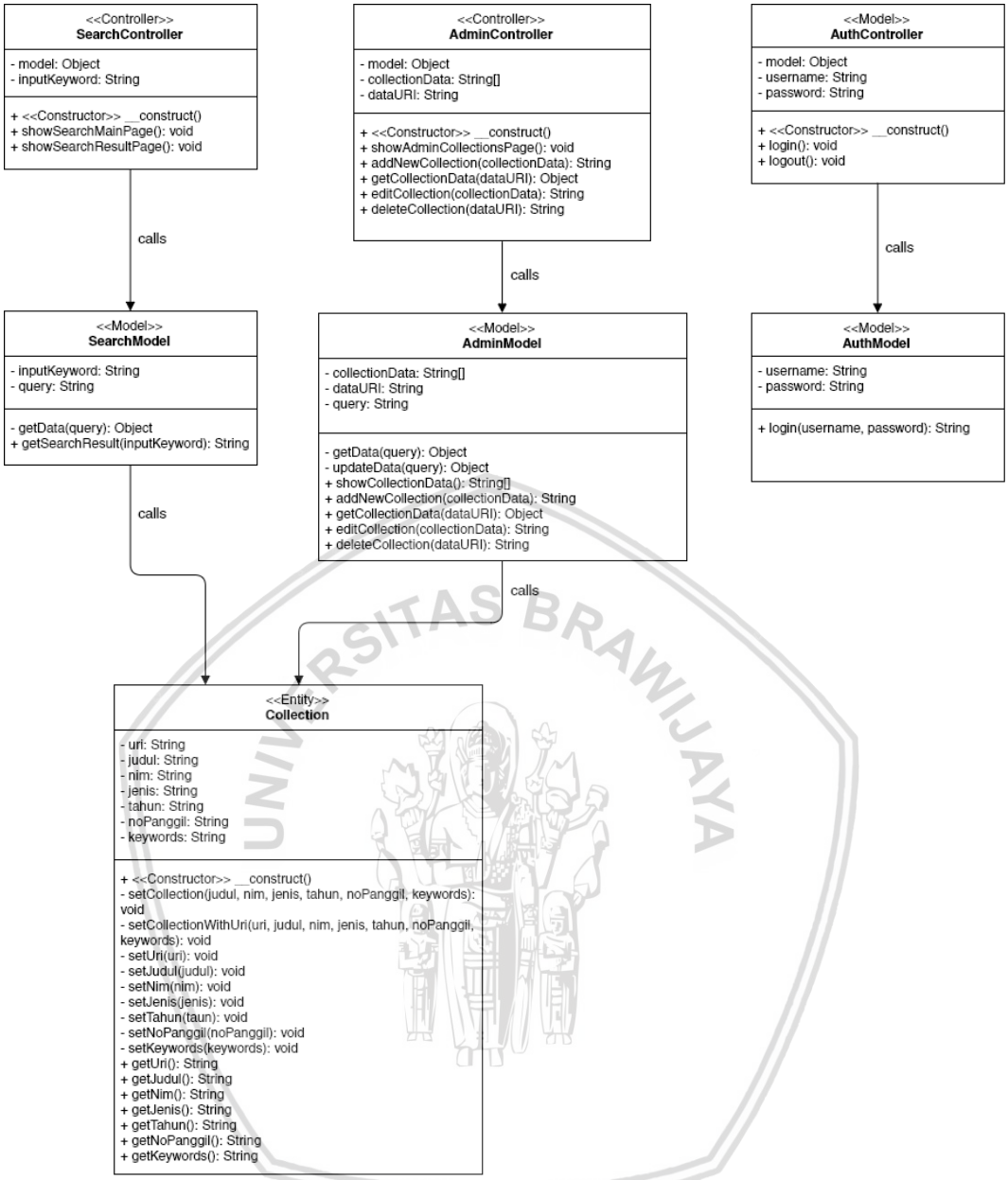


Gambar 5.5 *Sequence Diagram* Menghapus Data Koleksi

Gambar 5.5 menggambarkan interaksi aktor (pengunjung) yang melakukan aktifitas menghapus data koleksi. Pertama aktor menekan tombol *Delete* di halaman daftar koleksi maka sistem akan menampilkan halaman *popover* hapus data koleksi. Lalu aktor menekan tombol *Yes* untuk mengkonfirmasi proses hapus data koleksi. Kelas AdminController akan meneruskan informasi ke kelas AdminModel untuk melakPrasukan proses hapus data koleksi. Kemudian AdminModel akan mengembalikan konfirmasi hasil penyimpanan ke AdminController untuk kemudian ditampilkan.

5.1.2 Class Diagram

Perancangan *class diagram* pada sistem ini ditunjukkan pada Gambar 5.6.



Gambar 5.6 *Class Diagram*

5.2 Perancangan Data

Pada perancangan data untuk sistem pencarian koleksi dilakukan perancangan ontologi yang akan dirancang dengan menggunakan referensi dari sampel data yang didapat dari Ruang Baca FILKOM dengan penambahan untuk penyesuaian kebutuhan penelitian.

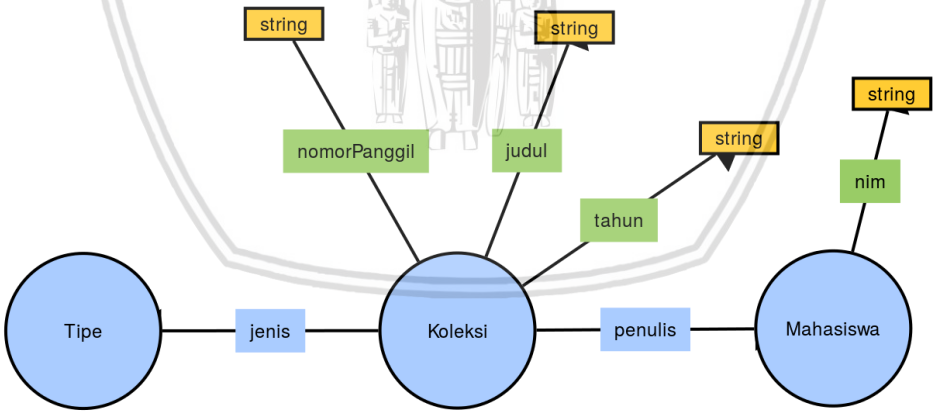
Dalam perancangan ontologi diperlukan rancangan model basis pengetahuan (*knowledge base*) atau ontologi sebelum di implementasi ke dalam *ontology language* (OWL). Terdapat tahapan dalam perancangan model ontologi, antara lain:

1. Menentukan domain ontologi yang dimana dalam penelitian ini adalah koleksi skripsi dan PKL di Ruang Baca FILKOM Universitas Brawijaya;
2. Menentukan batasan ontologi dalam struktur ontologi. Dalam ontologi yang akan dibuat sebuah koleksi terdiri atas judul, NIM penulis, jenis koleksi, tahun, dan nomor panggil;
3. Mendefinisikan *class* ontology dan menyusun *class* tersebut dalam hirarki (*superclass-subclass*) dengan mendefinisikan konsep umum dalam domain dilanjutkan dengan konsep yang lebih spesifik.

   Pada penelitian ini dilakukan penentuan *class* untuk domain koleksi skripsi dan PKL, yaitu:

   - Koleksi, merupakan *class* untuk mendefinisikan koleksi.
   - Mahasiswa, merupakan *class* untuk mendefinisikan penulis koleksi.
   - Tipe, merupakan *class* untuk mendefinisikan jenis koleksi.
4. Mendefinisikan *property* pada *class*. Pada *property* terdapat konstrain atau batasan tertentu dimana *property* yang dimiliki suatu *class* memiliki tipe nilai khusus. Terdapat dua kategori konstrain dalam pengembangan ontology, yaitu:

   - Kardinalitas, yaitu banyaknya nilai yang dimiliki suatu *property*; dan
   - Tipe, yaitu tipe data yang harus didefinisikan. Terdapat beberapa tipe data secara umum, diantaranya *string*, *number*, *boolean*, dan *instance*.



Gambar 5.7 Rancangan Struktur

5.2.1 Deskripsi Property pada Class Ontologi

Tabel 5.1 *Property Class* Koleksi

Property	**Kardinalitas**	***Type***	**Keterangan**
judul	Single	String	Judul koleksi
jenis	Single (Tipe)	Instance	Jenis koleksi

nomorPanggil	Single	String	Nomor panggil koleksi
penulis	Single (Mahasiswa)	Instance	Penulis koleksi
tahun	Single	String	Tahun koleksi

Tabel 5.2 *Property Class* Mahasiswa

Property	Kardinalitas	*Type*	Keterangan
nim	Single	String	NIM mahasiswa

5.3 Perancangan Komponen

Pada perancangan komponen dilakukan dekomposisi sub-sistem menjadi komponen detil. Perancangan komponen menjelaskan secara detil dari atribut dan algortima fungsi yang terdapat pada suatu *class* yang telah dimodelkan pada *class diagram*. Dalam perancangan komponen pada sistem ini memakai tiga sampel algoritme fungsi, yaitu algoritme fungsi tambah koleksi, algoritme fungsi menampilkan ambil data koleksi, dan algoritme fungsi hapus koleksi.

5.3.1 Algoritme Fungsi Tambah Koleksi

Algoritme fungsi tambah koleksi merupakan algoritme dari fungsi sistem untuk menambah koleksi. Fungsi ini merupakan bagian dari *class model* AdminModel dan bernama addNewCollection(). Kode Program 5.1 merupakan *pseudocode* dari algoritme fungsi tambah koleksi.

Kode Program 5.1 *Pseudocode* Algoritme Fungsi Tambah Koleksi

```
BEGIN addNewCollection(data)
        Declaration URI
        Initialization object
        Get object data
        Adding object data to query
        Do request to store with query, get response
        IF response == '200'
                return '200'
        ELSE
                return '400'
        END IF
END addNewCollection
```

5.3.2 Algoritme Fungsi Ambil Hasil Pencarian

Algoritme fungsi ambil koleksi merupakan algoritme dari fungsi sistem untuk mengambil hasil pencarian. Fungsi ini merupakan bagian dari *class model*

SearchModel dan bernama getSearchResult(). Kode Program 5.2 merupakan *pseudocode* dari algoritme fungsi ambil hasil pencarian.

Kode Program 5.2 *Pseudocode* Algoritme Fungsi Ambil Hasil Pencarian

```
BEGIN getSearchResult(q)
        IF q is not empty
                Adding q to query
                Do request to store with query, get response
                IF response is set
                        Set i = 0
                        FOREACH response
                                Initialization object
                                Get object data
                                i++
                        END FOREACH
                        return response
                ELSE
                        return false
                END IF
        ELSE
                return false
        ENDIF
END getSearchResult
```

5.3.3 Algoritme Fungsi Hapus Koleksi

Algoritme fungsi hapus koleksi merupakan algoritme dari fungsi sistem untuk menghapus koleksi. Fungsi ini merupakan bagian dari *class model* AdminModel dan bernama deleteCollection(). Kode Program 5.3 merupakan *pseudocode* dari algoritme fungsi hapus koleksi.

Kode Program 5.3 *Pseudocode* Algoritme Fungsi Hapus Koleksi

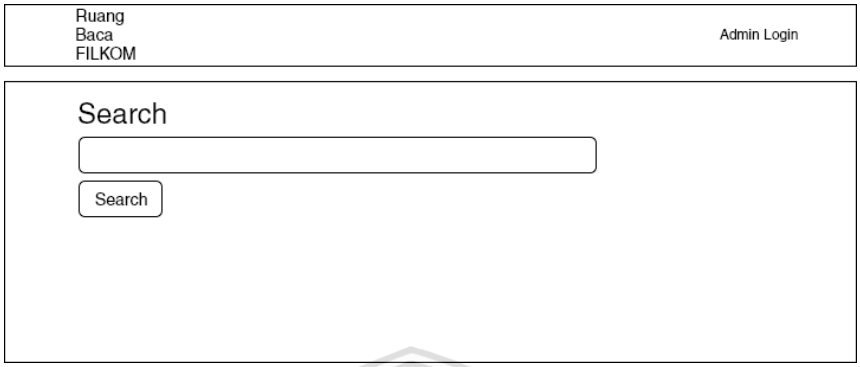
```
BEGIN deleteCollection(dataUri)
        Adding dataUri to query
        Do request to store with query, get response
        IF response == '200'
                return '200'
        ELSE
                return 'Failed to delete collection'
        END IF
END deleteCollection
```

5.4 Perancangan Antarmuka

Pada bagian ini akan menjelaskan perancangan antarmuka dari sistem pencarian koleksi.

5.4.1 Rancangan Antarmuka Halaman Utama Pengunjung

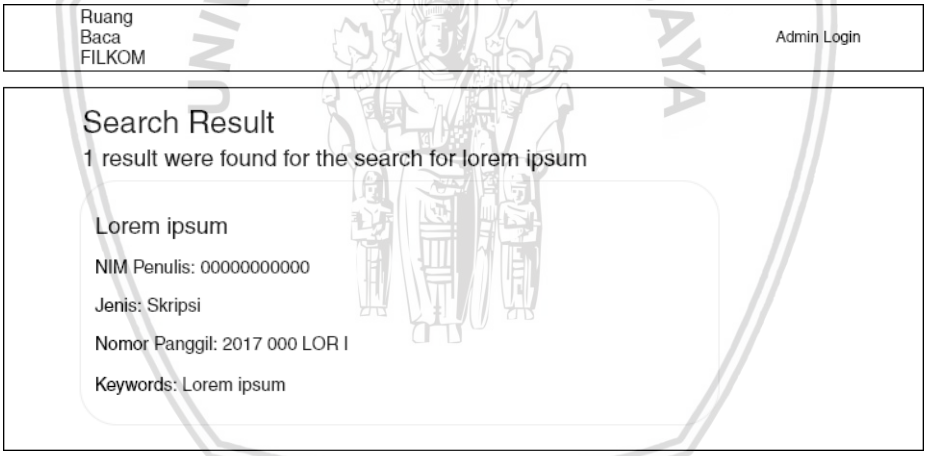
Halaman utama pengunjung merupakan halaman untuk melakukan pencarian koleksi. Rancangan antarmuka halaman utama dapat dilihat pada Gambar 5.7.



Gambar 5.8 Rancangan Antarmuka Halaman Utama Pengunjung

5.4.2 Rancangan Antarmuka Halaman Hasil Pencarian

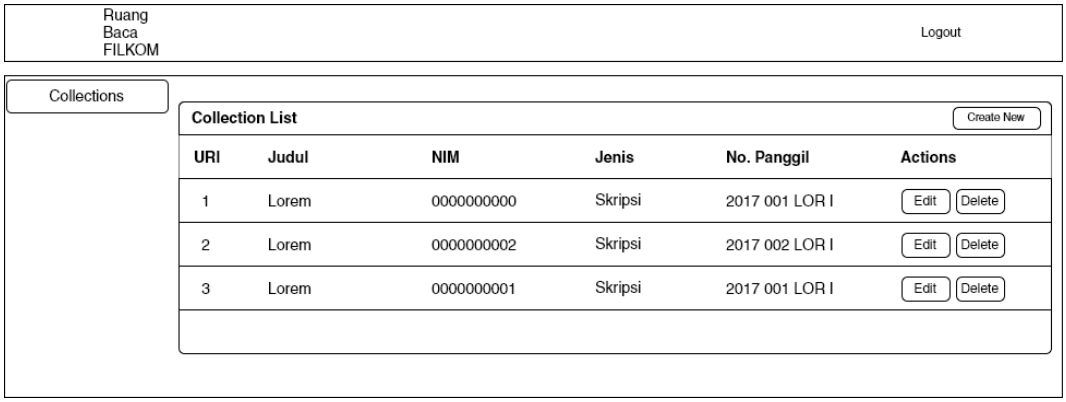
Halaman hasil pencarian merupakan halaman untuk menampilkan hasil pencarian. Rancangan antarmuka halaman hasil pencarian dapat dilihat pada Gambar 5.8.



Gambar 5.9 Rancangan Antarmuka Halaman Hasil Pencarian

5.4.3 Rancangan Antarmuka Halaman Daftar Koleksi

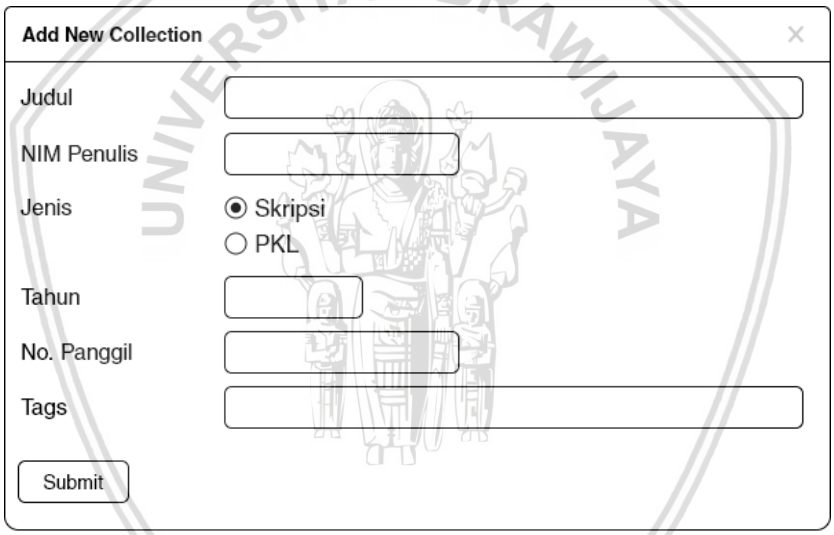
Halaman daftar koleksi merupakan halaman untuk menampilkan data koleksi yang terdapat pada ontologi. Rancangan antarmuka halaman daftar koleksi dapat dilihat pada Gambar 5.9.



Gambar 5.10 Rancangan Antarmuka Halaman Daftar Koleksi

5.4.4 Rancangan Antarmuka *Popup* Form Tambah Data Koleksi

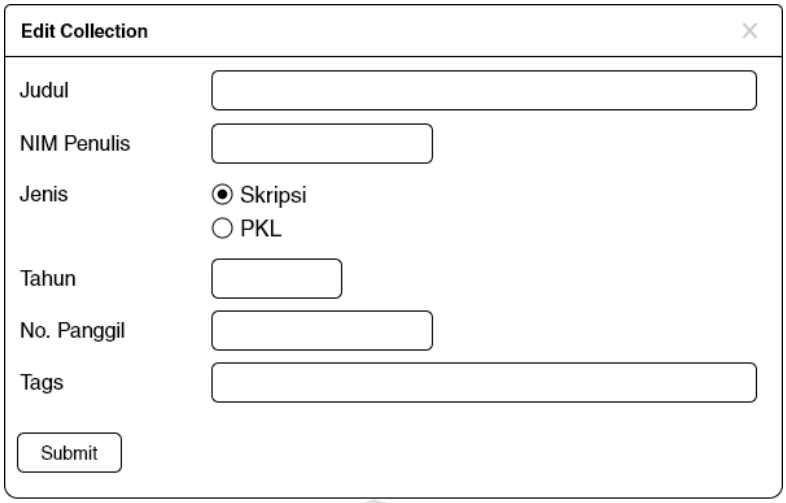
Popup form tambah data koleksi merupakan halaman *popup* untuk menampilkan form untuk menambah data koleksi baru. Rancangan antarmuka halaman hasil pencarian dapat dilihat pada Gambar 5.10.



Gambar 5.11 Rancangan Antarmuka *Popup* Form Tambah Data Koleksi

5.4.5 Rancangan Antarmuka *Popup* Form Ubah Data Koleksi

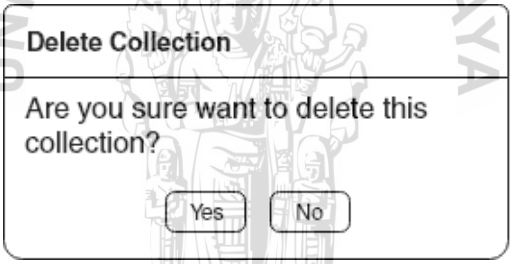
Popup form ubah data koleksi merupakan halaman *popup* untuk menampilkan form untuk mengubah data koleksi yang terdapat pada ontologi. Rancangan antarmuka halaman hasil pencarian dapat dilihat pada Gambar 5.11.



Gambar 5.12 Rancangan Antarmuka *Popup* Form Ubah Data Koleksi

5.4.6 Rancangan Antarmuka *Popover* Hapus Data Koleksi

Popover hapus data koleksi merupakan *popover* untuk menampilkan konfirmasi jika aktor ingin menghapus data koleksi yang terdapat pada ontologi. Rancangan antarmuka *popover* hapus data koleksi dapat dilihat pada Gambar 5.12.



Gambar 5.13 Rancangan Antarmuka *Popover* Hapus Data Koleksi

BAB 6 IMPLEMENTASI

6.1 Lingkungan Implementasi

Pada penelitian ini dibahas mengenai lingkungan implementasi yang terdiri dari lingkungan perangkat keras dan lingkungan perangkat lunak.

6.1.1 Lingkungan Perangkat Keras

Implementasi sistem pencarian koleksi ini dilakukan pada sebuah laptop dengan spesifikasi perangkat keras yang dapat dilihat pada Tabel 6.1.

Tabel 6.1 Spesifikasi Perangkat Keras

Nama Komponen	Spesifikasi Perangkat Keras
Prosesor	Intel Core i5-4210U 1,7 GHz
RAM	12 GB 1600 MHz DDR3
GPU	Intel HD Graphics Family + NVIDIA GeForce 840M

6.1.2 Lingkungan Perangkat Lunak

Implementasi sistem pencarian koleksi ini dilakukan pada *platform* perangkat lunak Windows. Adapun spesifikasi perangkat lunak yang terpasang pada laptop yang digunakan dalam penelitian ini dapat dilihat pada Tabel 6.2.

Tabel 6.2 Spesifikasi Perangkat Lunak

Nama Komponen	Spesifikasi Perangkat Lunak
Sistem operasi	Windows 10 64-bit
Text editor	Sublime Text 3
Bahasa pemrograman	PHP
Ontology editor	Protégé
Graph store server	Apache Jena Fuseki

6.2 Implementasi Kelas

Implementasi sistem pencarian koleksi dilakukan dengan menggunakan pola MVC. *Class diagram* yang telah dirancang sebelumnya direalisasikan ke dalam bentuk *class Model* dan *Controller*. Semua *class Model* dan *Controller* ditulis dengan menggunakan Bahasa pemrograman PHP dan disimpan dengan *file* berekstensi **.php*. Tabel 6.3 menjelaskan *class* dengan *file* yang dibuat.

Tabel 6.3 Implementasi Kode

No.	Tipe *class*	Nama *class*	Lokasi *file*	Nama *file*
1	*Controller*	SearchController	/inc/controllers	search.controller.php

2	*Controller*	AdminController	/inc/controllers	admin.controller.php
3	*Controller*	AuthController	/inc/controllers	auth.controller.php
4	*Model*	SearchModel	/inc/models	search.model.php
5	*Model*	AdminModel	/inc/models	admin.model.php
6	*Model*	AuthModel	/inc/models	auth.model.php
7	*Entity*	Collection	/inc/entities	collection.php

6.3 Implementasi Komponen

Berikut adalah hasil implementasi komponen yang telah dijelaskan pada bagian perancangan.

6.3.1 Implementasi Algoritme Fungsi Ambil Hasil Pencarian

Kode Program 6.1 Implementasi Algoritme Fungsi Tambah Koleksi

```
public function addNewCollection($data) {
        $koleksiUri = str_replace(' ', '_', implode(' ',
array_slice(str_word_count($data['judul'], 2), 0, 5)));
        $object = new Collection($koleksiUri, $data['judul'], $data['nim'],
$data['jenis'], $data['tahun'], $data['nopanggil']);
        $dataObj['uri'] = $object->getUri();
        $dataObj['judul'] = $object->getJudul();
        $dataObj['nim'] = $object->getNim();
        $dataObj['jenis'] = $object->getJenis();
        $dataObj['tahun'] = $object->getTahun();
        $dataObj['nomorPanggil'] = $object->getNoPanggil();
        $queryData = "
                INSERT DATA {
                        catalog:{$dataObj['uri']} rdf:type catalog:Koleksi ,
owl:NamedIndividual .
                        catalog:{$dataObj['uri']} catalog:jenis
catalog:{$dataObj['jenis']} .
                        catalog:{$dataObj['uri']} catalog:judul '{$dataObj['judul']}'
.
                        catalog:{$dataObj['uri']} catalog:nomorPanggil
'{$dataObj['nomorPanggil']}' .
                        catalog:{$dataObj['uri']} catalog:tahun '{$dataObj['tahun']}'
.
                        catalog:{$dataObj['uri']} catalog:penulis
catalog:{$dataObj['nim']} .
                        catalog:{$dataObj['nim']} rdf:type catalog:Mahasiswa ,
owl:NamedIndividual .
                        catalog:{$dataObj['nim']} catalog:nim '{$dataObj['nim']}'
        }";
        $responseData = $this->updateData($queryData);
        if ($responseData == '200') {
```

```
            return '200';
        } else {
            return '400';
        }
}
```

6.3.2 Implementasi Algoritme Fungsi Ambil Hasil Pencarian

Kode Program 6.2 Implementasi Algoritme Fungsi Ambil Hasil Pencarian

```
public function getSearchResult($q) {
        if ($q != '') {
            $query = "
            SELECT ?judul ?nim ?jenis ?tahun ?nomorPanggil
            WHERE {
                ?koleksi a ?subject .
                ?subject rdfs:label ?label .
                ?koleksi catalog:judul ?judul .
                ?koleksi catalog:jenis ?jenis .
                ?koleksi catalog:penulis ?mhs .
                ?mhs catalog:nim ?nim .
                ?koleksi catalog:tahun ?tahun .
                ?koleksi catalog:nomorPanggil ?nomorPanggil .
                FILTER ( regex(?judul, \"$q\", \"i\") || regex(?nim, \"$q\", \"i\")
|| regex(?tahun, \"$q\", \"i\") || regex(?nomorPanggil, \"$q\", \"i\"))
            }
            ORDER BY ?tahun ?judul";
            $resArray = json_decode($this->getData($query), true);
            $resNew = $resArray['results']['bindings'];
            if (isset($resNew[0]['judul']['value'])) {
                $i = 0;
                foreach ($resNew as $val) {
                    $object{$i} = new Collection($val["judul"]["value"],
$val["nim"]["value"], substr($val["jenis"]["value"], strpos($val["jenis"]["value"],
"#") + 1), $val["tahun"]["value"], $val["nomorPanggil"]["value"]);
                    $response{$i}['judul'] = $object{$i}->getJudul();
                    $response{$i}['nim'] = $object{$i}->getNim();
                    $response{$i}['jenis'] = $object{$i}->getJenis();
                    $response{$i}['tahun'] = $object{$i}->getTahun();
                    $response{$i}['nomorPanggil'] = $object{$i}->getNoPanggil();
                    $i++;
                }
                return $response;
            } else {
                return false;
            }
        } else {
```

```
            return false;
        }
    }
```

6.3.3 Implementasi Algoritme Fungsi Hapus Koleksi

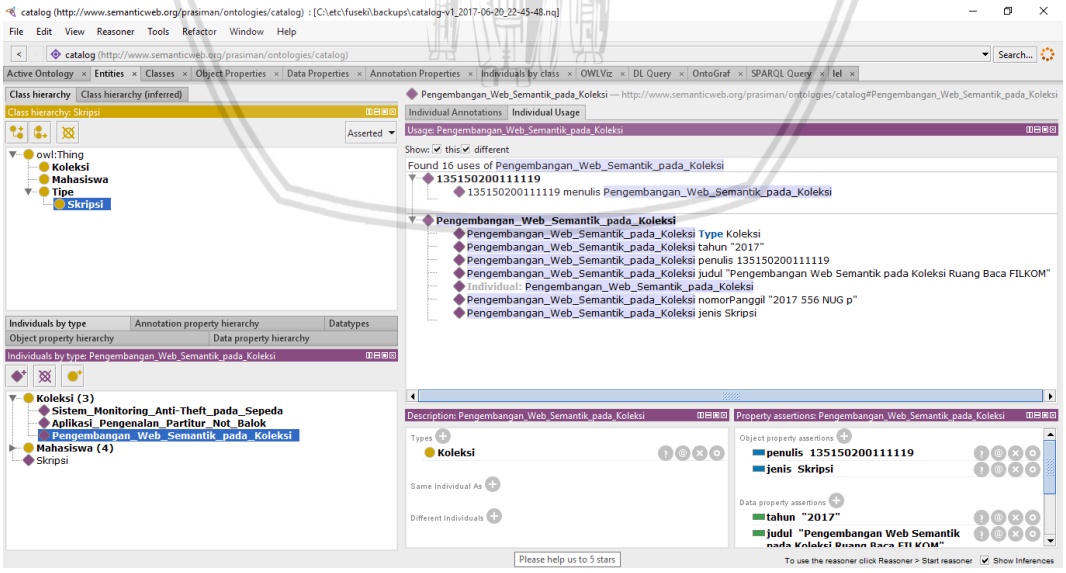
Kode Program 6.3 Implementasi Algoritme Fungsi Hapus Koleksi

```
public function deleteCollection($data) {
        $query = "
        DELETE WHERE {
            catalog:$data ?p ?o
        }";
        $delColResp = $this->updateData($query);
        if ($delColResp == '200') {
            return '200';
        } else {
            return 'Failed to delete collection';
        }
    }
```

6.4 Implementasi Ontologi

6.4.1 Implementasi Model Ontologi

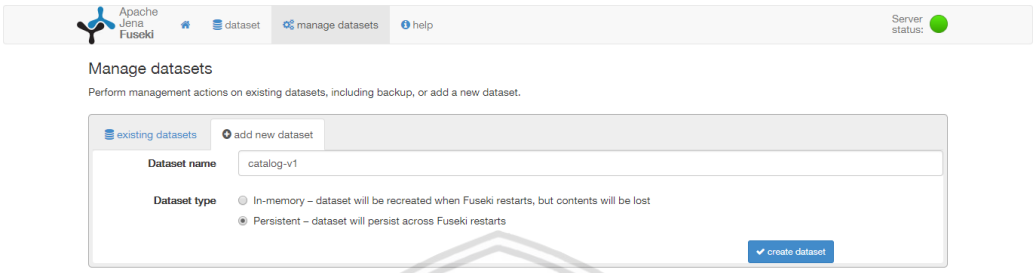
Ontologi yang sudah dirancang sebelumnya diimplementasikan didalam Protégé. Hasil dari implementasi tersebut kemudian di-*export*. Contoh pembuatan ontologi pada Protégé dapat dilihat pada Gambar 6.1.



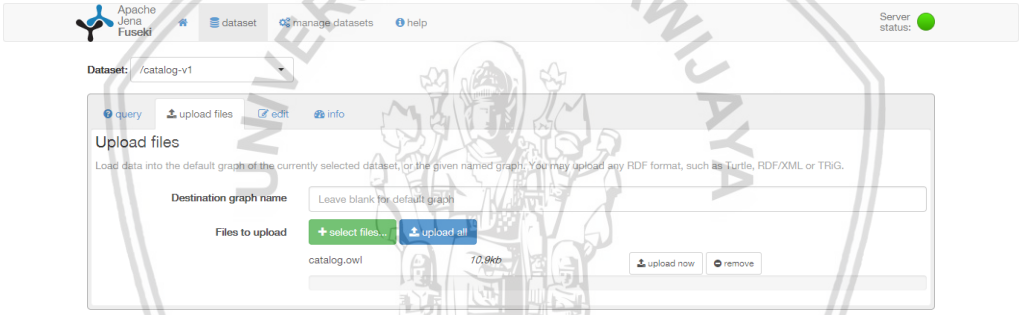
Gambar 6.1 Pembuatan Model Ontologi pada Protégé

6.4.2 Penyimpanan Ontologi

Ontologi yang sudah di-*export* dari Protégé kemudian di-*upload* ke dalam Apache Jena Fuseki. Sebelum ontologi dapat di-*upload,* terlebih dahulu sebuah *dataset* baru dibuat di dalam Apache Jena Fuseki. *Dataset* yang sudah dibuat kemudian digunakan untuk menyimpan *graph* atau ontologi yang sudah di-*export* sebelumnya pada Protégé.



Gambar 6.2 Pembuatan *Dataset* Baru pada Apache Jena Fuseki



Gambar 6.3 Proses *Upload* Ontologi ke *Dataset*

6.5 Implementasi Antarmuka

Pada bagian ini akan menjelaskan implementasi antarmuka dari sistem pencarian koleksi.

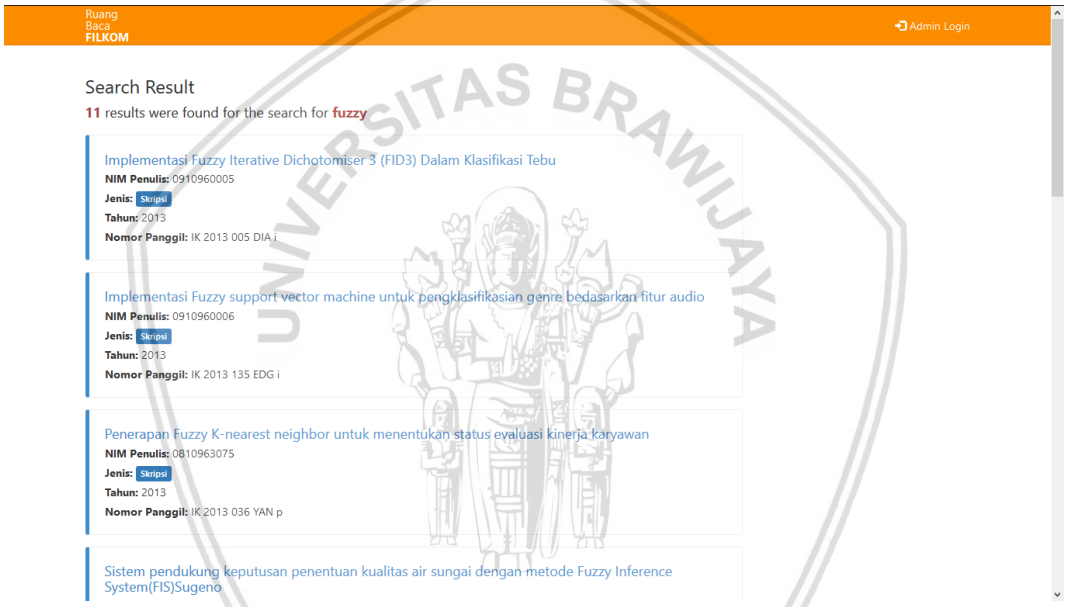
6.5.1 Antarmuka Halaman Utama Pengunjung

Halaman utama pengunjung merupakan halaman untuk melakukan pencarian koleksi untuk pengunjung. Untuk melakukan pencarian koleksi pengunjung harus mengisi kata kunci (*keyword*) yang diinginkan lalu menekan tombol *Search*. Jika tombol *Search* ditekan namun *search field* kosong maka sistem akan menampilkan notifikasi bahwa pengunjung belum mengisi *search field*. Antarmuka halaman utama dijelaskan pada Gambar 6.4.

Gambar 6.4 Antarmuka Halaman Utama Pengunjung

6.5.2 Antarmuka Halaman Hasil Pencarian

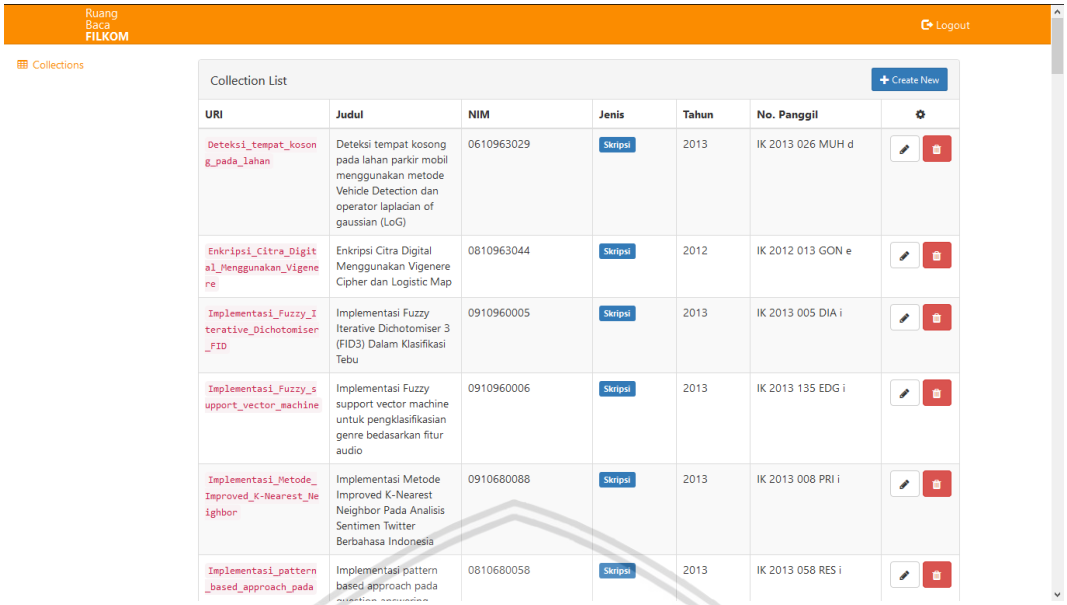
Halaman hasil pencarian merupakan halaman untuk menampilkan hasil pencarian untuk pengunjung. Hasil pencarian didasarkan pada *keyword* yang sudah dimasukkan oleh aktor di halaman utama pengunjung. Antarmuka halaman hasil pencarian dijelaskan pada Gambar 6.5.



Gambar 6.5 Antarmuka Halaman Hasil Pencarian

6.5.3 Antarmuka Halaman Daftar Koleksi

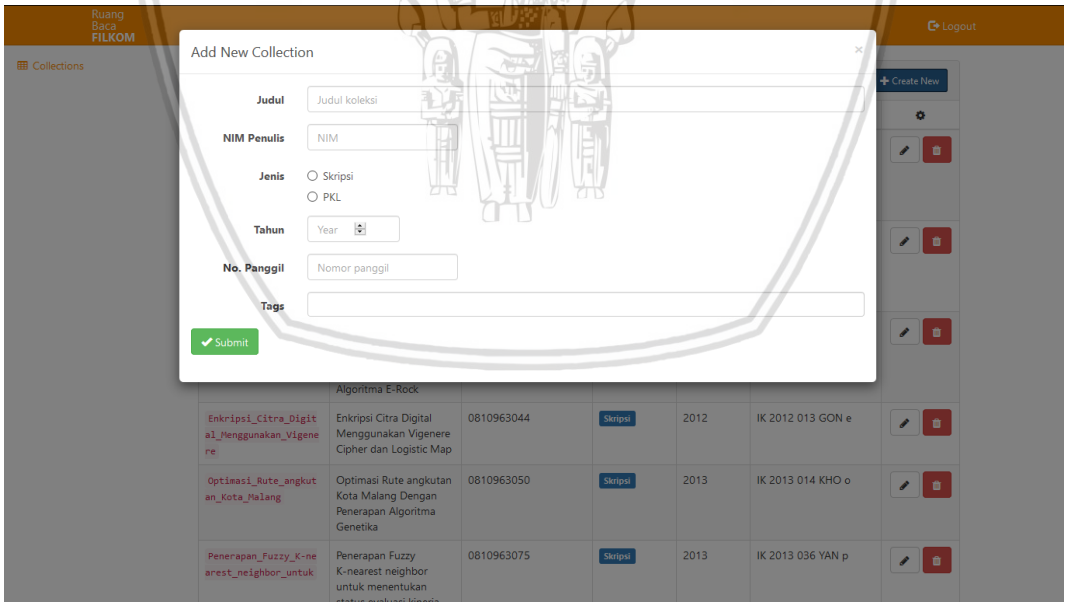
Halaman daftar koleksi merupakan halaman untuk menampilkan data koleksi yang ada. Halaman ini hanya dapat diakses oleh admin. Halaman ini berisi daftar koleksi yang dimana datanya ditampilkan pada tabel dan dapat diperbarui atau dihapus oleh admin. Pada tabel koleksi terdapat kolom yaitu *URI* data, judul, NIM penulis, tahun koleksi, nomor panggil koleksi, dan opsi untuk mengubah dan menghapus data koleksi. Admin dapat menambah data koleksi baru dengan menekan tombol *Add New* untuk menampilkan *popup* form tambah data koleksi. Untuk memperbarui atau menghapus data koleksi aktor dapat menekan tombol dengan ikon di pojok kanan tabel. Tombol dengan ikon pensil digunakan untuk menampilkan *popover* form ubah data koleksi, sedangkan tombol dengan ikon tempat sampah digunakan untuk menampilkan *popover* hapus data koleksi. Antarmuka halaman daftar koleksi dijelaskan pada Gambar 6.6.



Gambar 6.6 Antarmuka Halaman Daftar Koleksi

6.5.4 Antarmuka *Popup* Form Tambah Data Koleksi

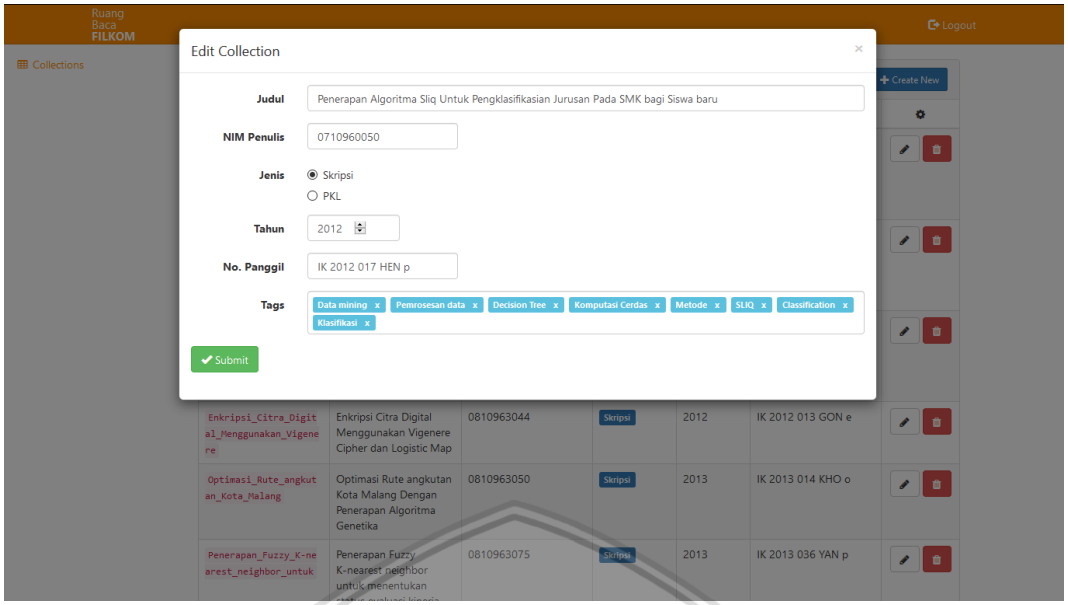
Popup form tambah data koleksi merupakan halaman *popup* untuk menampilkan form untuk menambah data koleksi baru. *Popup* ini muncul ketika admin menekan tombol *Add New*. Antarmuka halaman hasil pencarian dijelaskan pada Gambar 6.7.



Gambar 6.7 Antarmuka *Popup* Form Tambah Data Koleksi

6.5.5 Antarmuka *Popup* Form Ubah Data Koleksi

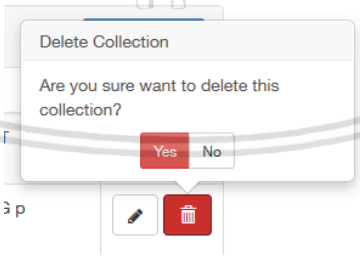
Popup form ubah koleksi merupakan halaman untuk menampilkan form untuk mengubah data koleksi yang ada. Antarmuka halaman hasil pencarian dijelaskan pada Gambar 6.8.



Gambar 6.8 Antarmuka *Popup* Form Ubah Data Koleksi

6.5.6 Antarmuka *Popover* Hapus Data Koleksi

Popover hapus data koleksi merupakan antarmuka untuk menampilkan *popover* untuk konfirmasi penghapusan data koleksi kepada pengguna. *Popover* ini muncul jika pengguna menekan tombol warna merah dengan ikon tempat sampah pada kolom opsi. Jika pengguna menekan tombol *Yes*, sistem akan menampilkan notifikasi bahwa data koleksi berhasil dihapus dan sistem akan memperbarui halaman daftar koleksi. Apabila pengguna memilih *No* pada halaman *popover* konfirmasi maka sistem akan menutup halaman *popover* dan proses penghapusan data koleksi dibatalkan. Antarmuka *popover* hapus data koleksi dijelaskan pada Gambar 6.9.



Gambar 6.9 Antarmuka *Popover* Hapus Data Koleksi

BAB 7 PENGUJIAN

7.1 Pengujian Unit

Proses pengujian *basis path* merupakan pengujian unit yang dilakukan dengan menggunakan *pseudocode* yang menerapkan fungsi atau logika yang digunakan oleh sistem dalam pencarian koleksi. *Pseudocode* tersebut akan dimodelkan ke dalam *flow graph*. Proses ini dilakukan untuk menentukan jumlah *cyclomatic complexity* (kompleksitas siklomatis) dan menentukan jalur independen. Jumlah kompleksitas siklomatis atau V(G) diperoleh dengan tiga cara berikut:

1. V(G) = E - N + 2, dimana

   E : Sisi atau *edge* (garis penghubung antar *node*);

   N : Jumlah simpul (*node*);

2. V(G) = P + 1, dimana

   P : *Predicate node* pada *flow graph*; dan

3. V(G) = R, dimana

   R : Jumlah *region* pada *flow graph*.

Algoritme yang akan diuji merupakan algoritme fungsi tambah koleksi, algoritme fungsi ambil koleksi, dan algoritme fungsi hapus koleksi.

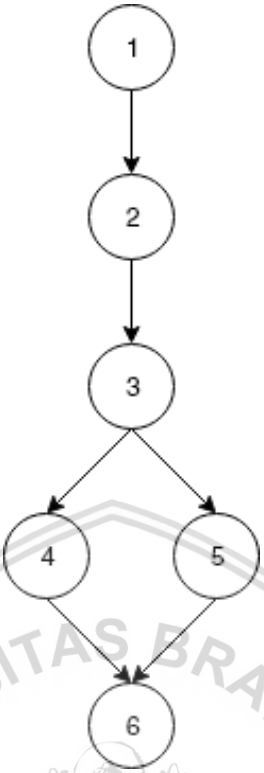
7.1.1 Pengujian Algoritme Fungsi Tambah Koleksi

Pseudocode yang akan digunakan dalam menggambarkan *flow graph* dari algoritme fungsi tambah koleksi dijelaskan pada Tabel 7.1.

Tabel 7.1 *Pseudocode* Algoritme Fungsi Tambah Koleksi

Pseudocode	Node
BEGIN addNewCollection(data)	1
Declaration URI	2
Initialization object	
Get object data	
Adding object data to query	
Do request to store with query, get response	
IF response == '200'	3
return '200'	4
ELSE	
return '400'	5
END IF	
END addNewCollection	6

Berdasarkan *pseudocode* dalam Tabel 7.1 dihasilkan *flow chart* yang dapat dilihat pada Gambar 7.1.



Gambar 7.1 *Flow Graph* Algoritme Fungsi Tambah Koleksi

Berdasarkan *flow graph* yang diperoleh pada Gambar 7.1 maka diperoleh *cyclomatic complexity* sebagai berikut:

1. V(G) = E - N + 2
   = 6 - 6 + 2
   = 2
2. V(G) = P + 1
   = 1 + 1
   = 2
3. V(G) = Jumlah *region*
   = 2

Berdasarkan jumlah *cyclomatic complexity* yang telah didapatkan, maka didapatkan *independent path* yaitu:

Jalur 1 : 1 – 2 – 3 – 4 – 6

Jalur 2 : 1 – 2 – 3 – 5 – 6

Berdasarkan *independent path* diatas, maka diperoleh kasus uji yang dijelaskan pada Tabel 7.2.

Tabel 7.2 Kasus Uji Algoritme Fungsi Tambah Koleksi

Jalur	Data Input	Hasil yang Diharapkan	Hasil yang Diperoleh	Status
1	judul => “Aplikasi Web Semantik” nim => “135150200111119” jenis => “Skripsi” tahun => “2017” nomorPanggil => “S-IK 2017 666 APL p”	return ‘200’	return ‘200’	Valid
2	judul => “;” nim => “;” jenis => “;” tahun => “;” nomorPanggil => “;”	return ’400’	return ’400’	Valid

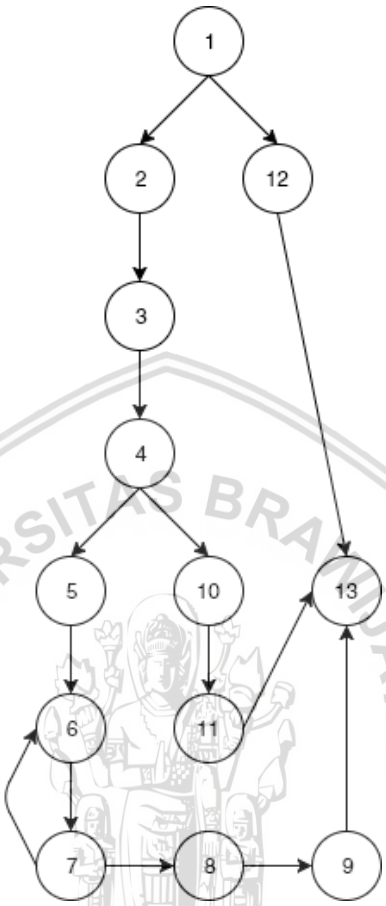
7.1.2 Pengujian Algoritme Fungsi Ambil Hasil Pencarian

Pseudocode yang akan digunakan dalam menggambarkan *flow graph* dari algoritme fungsi ambil pencarian dijelaskan pada Tabel 7.3.

Tabel 7.3 *Pseudocode* Algoritme Fungsi Ambil Hasil Pencarian

Pseudocode	Node
BEGIN getSearchResult(q)	1
IF q is not empty	2
Adding q to query	3
Do request to store with query, get response	
IF response is set	4
Set i = 0	5
FOREACH response	6
Initialization object	7
Get object data	
i++	
END FOREACH	8
return response	9
ELSE	
return false	10
END IF	11
ELSE	
return false	12
ENDIF	
END getSearchResult	13

Berdasarkan *pseudocode* dalam Tabel 7.3 dihasilkan *flow chart* yang dapat dilihat pada Gambar 7.2.



Gambar 7.2 *Flow Graph* Algoritme Fungsi Ambil Hasil Pencarian

Berdasarkan *flow graph* yang diperoleh pada Gambar 7.2 maka diperoleh *cyclomatic complexity* sebagai berikut:

1. V(G) = E - N + 2
   = 16 - 14 + 2
   = 4
2. V(G) = P + 1
   = 3 + 1
   = 4
3. V(G) = Jumlah *region*
   = 4

Berdasarkan jumlah *cyclomatic complexity* yang telah didapatkan, maka didapatkan *independent path* yaitu:

Jalur 1 : 1 – 12 – 13

Jalur 2 : 1 – 2 – 3 – 4 – 10 – 11 – 13

Jalur 3 : 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 13

Jalur 4 : 1 – 2 – 3 – 4 – 5 – 6 – 7 – 6 – 7 – 8 – 9 – 13

Berdasarkan *independent path* diatas, maka diperoleh kasus uji yang dijelaskan pada Tabel 7.4.

Tabel 7.4 Kasus Uji Algoritme Fungsi Ambil Hasil Pencarian

Jalur	Data Input	Hasil yang Diharapkan	Hasil yang Diperoleh	Status
1	q => null	return false	return false	Valid
2	q => “web semantik”	return false	return false	Valid
3	q => “bahasa inggris”	return Array ([0] => Array ([judul] => Klasifikasi Jurnal ilmiah berbahasa inggris berdasarkan abstrak menggunakan algoritma ID3 [nim] => 0710960028 [jenis] => Skripsi [tahun] => 2012 [nomorPanggil] => IK 2012 027 BUD k))	return Array ([0] => Array ([judul] => Klasifikasi Jurnal ilmiah berbahasa inggris berdasarkan abstrak menggunakan algoritma ID3 [nim] => 0710960028 [jenis] => Skripsi [tahun] => 2012 [nomorPanggil] => IK 2012 027 BUD k))	Valid
4	q => “citra digital”	return Array ([0] => Array ([judul] => Enkripsi Citra Digital Menggunakan Vigenere Cipher dan Logistic Map [nim] => 0810963044 [jenis] => Skripsi [tahun] => 2012 [nomorPanggil] => IK 2012 013 GON e) [1] => Array ([judul] => sistem deteksi slot parkir menggunakan pengolahan citra digital dengan metode thresholding [nim] => 115060900111018 [jenis] => Skripsi [tahun] => 2015 [nomorPanggil] => S-IK 2015 238 MUH s))	return Array ([0] => Array ([judul] => Enkripsi Citra Digital Menggunakan Vigenere Cipher dan Logistic Map [nim] => 0810963044 [jenis] => Skripsi [tahun] => 2012 [nomorPanggil] => IK 2012 013 GON e) [1] => Array ([judul] => sistem deteksi slot parkir menggunakan pengolahan citra digital dengan metode thresholding [nim] => 115060900111018 [jenis] => Skripsi [tahun] => 2015 [nomorPanggil] => S-IK 2015 238 MUH s))	Valid

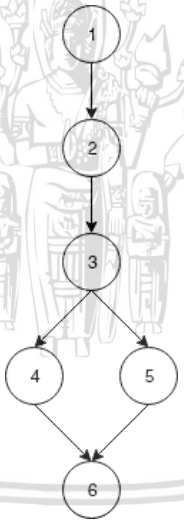
7.1.3 Pengujian Algoritme Fungsi Hapus Koleksi

Pseudocode yang akan digunakan dalam menggambarkan *flow graph* dari algoritme fungsi hapus koleksi dijelaskan pada Tabel 7.4.

Tabel 7.5 *Pseudocode* Algoritme Fungsi Hapus Koleksi

Pseudocode	Node
BEGIN deleteCollection(dataUri)	1
Adding dataUri to query	2
Do request to store with query, get response	
IF response == '200'	3
return '200'	4
ELSE	
return 'Failed to delete collection'	5
END IF	
END deleteCollection	6

Berdasarkan *pseudocode* dalam Tabel 7.5 dihasilkan *flow chart* yang dapat dilihat pada Gambar 7.3.



Gambar 7.3 *Flow Graph* Algoritme Fungsi Hapus Koleksi

Berdasarkan *flow graph* yang diperoleh pada Gambar 7.2 maka diperoleh *cyclomatic complexity* sebagai berikut:

1. V(G) = E - N + 2
   = 6 - 6 + 2
   = 2
2. V(G) = P + 1
   = 1 + 1
   = 2

3. V(G) = Jumlah *region*

= 2

Berdasarkan jumlah *cyclomatic complexity* yang telah didapatkan, maka didapatkan *independent path* yaitu:

Jalur 1 : 1 – 2 – 3 – 4 – 6

Jalur 2 : 1 – 2 – 3 – 5 – 6

Berdasarkan *independent path* diatas, maka diperoleh kasus uji yang dijelaskan pada Tabel 7.6.

Tabel 7.6 Kasus Uji Algoritme Fungsi Hapus Koleksi

Jalur	Data Input	Hasil yang Diharapkan	Hasil yang Diperoleh	Status
1	koluri => 'Aplikasi_Web_Semantik'	return '200'	return '200'	Valid
2	koluri => 'Aplikasi Web Semantik'	return 'Failed to delete collection'	return 'Failed to delete collection'	Valid

7.2 Pengujian Validasi

Pengujian validasi dilakukan dengan menggunakan metode *black box* untuk memastikan bahwa seluruh kebutuhan fungsional dari sistem berjalan sesuai dengan yang diharapkan. Berikut merupakan pengujian validasi berdasarkan scenario *use case* yang telah dibuat pada perancangan sistem.

7.2.1 Kebutuhan Fungsional RBF-F-001

Berikut ini adalah kasus uji dalam kebutuhan fungsional mencari informasi koleksi dengan kode kebutuhan RBF-F-001 yang dijelaskan pada Tabel 7.7 dan Tabel 7.8.

Tabel 7.7 Kasus Uji Validasi Alur Utama RBF-F-001

Nomor Kasus Uji	VAL-RBF-001
Nama Kasus Uji	Mencari informasi koleksi
Objek Uji	Kebutuhan fungsional RBF-F-001 (alur utama)
Tujuan	Untuk memastikan aktor (pengunjung) dapat melakukan pencarian informasi koleksi
Prosedur Uji	1. Aktor mengakses halaman utama pengunjung. 2. Aktor memasukkan *keyword* pada *search field*. 3. Aktor menekan tombol *Search*.
Hasil yang Diharapkan	Sistem akan menampilkan informasi hasil pencarian koleksi.

Tabel 7.8 Kasus Uji Validasi Alur Alternatif RBF-F-001

Nomor Kasus Uji	VAL-RBF-002
Nama Kasus Uji	Mencari informasi koleksi
Objek Uji	Kebutuhan fungsional RBF-F-001 (alur alternatif)
Tujuan	Untuk memastikan aktor (pengunjung) tidak dapat melakukan pencarian informasi koleksi
Prosedur Uji	1. Aktor mengakses halaman utama pengunjung. 2. Aktor mengosongkan *search field*. 3. Aktor menekan tombol *Search*.
Hasil yang Diharapkan	Sistem akan mengirim notifikasi bahwa kolom *search field* kosong dan aktor tidak akan diarahkan ke halaman hasil pencarian.

7.2.2 Kebutuhan Fungsional RBF-F-002

Berikut ini adalah kasus uji dalam kebutuhan fungsional menambah data koleksi dengan kode kebutuhan RBF-F-002 yang dijelaskan pada Tabel 7.9 dan Tabel 7.10.

Tabel 7.9 Kasus Uji Validasi Alur Utama RBF-F-002

Nomor Kasus Uji	VAL-RBF-003
Nama Kasus Uji	Menambah data koleksi
Objek Uji	Kebutuhan fungsional RBF-F-002 (alur utama)
Tujuan	Untuk memastikan aktor (admin) dapat melakukan penambahan data koleksi baru.
Prosedur Uji	1. Aktor mengakses halaman daftar koleksi. 2. Aktor menekan tombol *Add New* untuk menampilkan halaman *popup* form tambah koleksi baru. 3. Aktor mengisi seluruh *field* data yang dibutuhkan pada form. 4. Aktor menekan tombol *Submit*.
Hasil yang Diharapkan	Sistem akan menampilkan notifikasi bahwa koleksi baru berhasil ditambahkan dan sistem akan memperbarui halaman daftar koleksi.

Tabel 7.10 Kasus Uji Validasi Alur Alternatif RBF-F-002

Nomor Kasus Uji	VAL-RBF-004
Nama Kasus Uji	Menambah data koleksi

Objek Uji	Kebutuhan fungsional RBF-F-002 (alur alternatif)
Tujuan	Untuk memastikan aktor (admin) tidak dapat melakukan penambahan data koleksi baru.
Prosedur Uji	5. Aktor mengakses halaman daftar koleksi. 6. Aktor menekan tombol *Add New* untuk menampilkan halaman *popup* form tambah koleksi baru. 7. Aktor mengisi beberapa *field* data yang dibutuhkan pada form. 8. Aktor menekan tombol *Submit*.
Hasil yang Diharapkan	Sistem akan menampilkan notifikasi bahwa terdapat *field* kosong dan sistem tidak akan melakukan apapun.

7.2.3 Kebutuhan Fungsional RBF-F-003

Berikut ini adalah kasus uji dalam kebutuhan fungsional melihat data koleksi dengan kode kebutuhan RBF-F-003 yang dijelaskan pada Tabel 7.11.

Tabel 7.11 Kasus Uji Validasi Alur Utama RBF-F-003

Nomor Kasus Uji	VAL-RBF-005
Nama Kasus Uji	Melihat data koleksi
Objek Uji	Kebutuhan fungsional RBF-F-003 (alur utama)
Tujuan	Untuk memastikan aktor (admin) dapat melihat data koleksi.
Prosedur Uji	1. Aktor mengakses halaman daftar koleksi.
Hasil yang Diharapkan	Sistem akan menampilkan halaman daftar koleksi.

7.2.4 Kebutuhan Fungsional RBF-F-004

Berikut ini adalah kasus uji dalam kebutuhan fungsional mengubah data koleksi dengan kode kebutuhan RBF-F-004 yang dijelaskan pada Tabel 7.12 dan Tabel 7.13.

Tabel 7.12 Kasus Uji Validasi Alur Utama RBF-F-004

Nomor Kasus Uji	VAL-RBF-006
Nama Kasus Uji	Mengubah data koleksi
Objek Uji	Kebutuhan fungsional RBF-F-004 (alur utama)
Tujuan	Untuk memastikan aktor (admin) dapat melakukan perubahan pada data koleksi.
Prosedur Uji	1. Aktor mengakses halaman daftar koleksi.

	2. Aktor menekan tombol *Edit* untuk menampilkan halaman *popup* form ubah koleksi baru. 3. Aktor mengisi seluruh *field* data yang dibutuhkan pada form. 4. Aktor menekan tombol *Submit*.
Hasil yang Diharapkan	Sistem akan menampilkan notifikasi bahwa koleksi baru berhasil diperbarui dan sistem akan memperbarui halaman daftar koleksi.

Tabel 7.13 Kasus Uji Validasi Alur Alternatif RBF-F-004

Nomor Kasus Uji	VAL-RBF-007
Nama Kasus Uji	Mengubah data koleksi
Objek Uji	Kebutuhan fungsional RBF-F-004 (alur alternatif)
Tujuan	Untuk memastikan aktor (admin) tidak dapat melakukan perubahan pada data koleksi.
Prosedur Uji	1. Aktor mengakses halaman daftar koleksi. 2. Aktor menekan tombol *Edit* untuk menampilkan halaman *popup* form ubah koleksi baru. 3. Aktor mengisi beberapa *field* data yang dibutuhkan pada form. 4. Aktor menekan tombol *Submit*.
Hasil yang Diharapkan	Sistem akan menampilkan notifikasi bahwa terdapat *field* kosong dan sistem tidak akan melakukan apapun.

7.2.5 Kebutuhan Fungsional RBF-F-005

Berikut ini adalah kasus uji dalam kebutuhan fungsional menghapus data koleksi dengan kode kebutuhan RBF-F-005 yang dijelaskan pada Tabel 7.14 dan Tabel 7.15.

Tabel 7.14 Kasus Uji Validasi Alur Utama RBF-F-005

Nomor Kasus Uji	VAL-RBF-008
Nama Kasus Uji	Menghapus data koleksi
Objek Uji	Kebutuhan fungsional RBF-F-005 (alur utama)
Tujuan	Untuk memastikan aktor (admin) dapat menghapus data koleksi.
Prosedur Uji	1. Aktor mengakses halaman daftar koleksi. 2. Aktor menekan tombol *Delete* pada kolom *Action* di dalam tabel koleksi untuk menampilkan tampilan *popover* berisi konfirmasi untuk menghapus data.

	3. Aktor menekan tombol *Yes* pada tampilan *popover*.
Hasil yang Diharapkan	Sistem akan menampilkan notifikasi bahwa data koleksi berhasil dihapus dan sistem akan memperbarui halaman daftar koleksi.

Tabel 7.15 Kasus Uji Validasi Alur Alternatif RBF-F-005

Nomor Kasus Uji	VAL-RBF-009
Nama Kasus Uji	Mengubah data koleksi
Objek Uji	Kebutuhan fungsional RBF-F-005 (alur alternatif)
Tujuan	Untuk memastikan aktor (admin) tidak dapat menghapus data koleksi.
Prosedur Uji	1. Aktor mengakses halaman daftar koleksi. 2. Aktor menekan tombol *Delete* pada kolom *Action* di dalam tabel koleksi untuk menampilkan tampilan *popover* berisi konfirmasi untuk menghapus data. 3. Aktor menekan tombol *No* pada tampilan *popover*.
Hasil yang Diharapkan	Sistem akan menutup halaman *popover* dan proses hapus data koleksi dibatalkan.

7.3 Pengujian Perbandingan Pencarian Sintaksis dan Semantik

Pada pengujian perbandingan sintaksis dan semantik terdapat dua bagian pengujian. Tiap bagian akan dijelaskan pada subbab berikut.

7.3.1 Pengujian Perbandingan Pencarian Sintaksis dan Semantik Bagian I

Pada bagian ini dilakukan pengujian dengan melakukan pencarian pada data koleksi. Pada pengujian ini digunakan kata kunci sebagai kasus uji berupa informasi atau *field* yang terdapat pada data koleksi, misalnya judul, NIM penulis, tahun, dan nomor panggil. Dari tiap kasus uji akan dilakukan pencarian dengan menggunakan kata kunci yang diberikan terhadap *field* yang dituju. Pengujian ini dilakukan untuk menguji sistem dalam melakukan pencarian kata pada suatu *field* yang ditujukan. Hasil pengujian ini dapat dilihat pada Tabel 7.16.

Tabel 7.16 Hasil Pengujian Perbandingan Pencarian Sintaksis dan Semantik Bagian I

Kata kunci	*Field* target	Hasil yang diharapkan	Hasil yang didapatkan	Status
Decision support	Judul	1 koleksi dengan judul "*decision support*"	1 koleksi dengan judul "*decision support*"	Valid

Citra digital		2 koleksi dengan judul "citra digital"	2 koleksi dengan judul "citra digital"	Valid
Pakan ternak		1 koleksi dengan judul "pakan ternak"	1 koleksi dengan judul "pakan ternak"	Valid
Sistem pendukung keputusan		5 koleksi dengan judul "sistem pendukung keputusan"	5 koleksi dengan judul "sistem pendukung keputusan"	Valid
Sistem pakar		4 koleksi dengan judul "sistem pakar"	4 koleksi dengan judul "sistem pakar"	Valid
115060800111101	NIM penulis	1 koleksi dengan NIM penulis "115060800111101"	1 koleksi dengan NIM penulis "115060800111101"	Valid
0810680007		1 koleksi dengan NIM penulis "0810680007"	1 koleksi dengan NIM penulis "0810680007"	Valid
105090600111049		1 koleksi dengan NIM penulis "105090600111049"	1 koleksi dengan NIM penulis "105090600111049"	Valid
105060801111043		1 koleksi dengan NIM penulis "105060801111043"	1 koleksi dengan NIM penulis "105060801111043"	Valid
115060800111085		1 koleksi dengan NIM penulis "115060800111085"	1 koleksi dengan NIM penulis "115060800111085"	Valid
2013	Tahun	13 koleksi dengan tahun 2013	13 koleksi dengan tahun 2013	Valid
2014		15 koleksi dengan tahun 2014	15 koleksi dengan tahun 2014	Valid

2015		26 koleksi dengan tahun 2015	26 koleksi dengan tahun 2015	Valid
IK 2013 125 FRE r	Nomor panggil	1 koleksi dengan nomor panggil "IK 2013 125 FRE r"	1 koleksi dengan nomor panggil "IK 2013 125 FRE r"	Valid
S-IK 2015 095 MAR p		1 koleksi dengan nomor panggil "S-IK 2015 095 MAR p "	1 koleksi dengan nomor panggil "S-IK 2015 095 MAR p "	Valid

7.3.2 Pengujian Perbandingan Pencarian Sintaksis dan Semantik Bagian II

Pada bagian ini dilakukan pengujian untuk mengetahui perbandingan hasil pencarian berbasis sintaksis (kata kunci) yang mengacu pada judul yang terdapat pada suatu koleksi dengan hasil pencarian berbasis semantik yang mengacu pada kelompok topik pada koleksi. Pengujian ini dilakukan pada 52 data judul koleksi dan menggunakan 10 subjek atau topik sebagai kasus uji. Topik yang digunakan sebagai kasus uji dipilih berdasarkan data yang tersedia pada ontologi. Dari hasil pencarian dari masing-masing teknik akan dilakukan perhitungan jumlah data relevan diterima yang benar. Kemudian jumlah tersebut akan dikali 100% untuk mendapatkan nilai akurasi dalam bentuk persentase. Nilai akurasi tersebut menunjukkan tingkat keakuratan dari hasil pencarian kata kunci berdasarkan subjek tertentu. Hasil pengujian akurasi dapat dilihat pada Tabel 7.17.

Tabel 7.17 Hasil Pengujian Perbandingan Pencarian Sintaksis dan Semantik Bagian II

Subjek	Jumlah data yang didapatkan dari pencarian berbasis kata kunci	Jumlah data yang didapatkan dari pencarian berbasis semantik
Logika fuzzy	0 / 11 (0%)	11 / 11 (100%)
Citra digital	2 / 5 (40%)	5 / 5 (100%)
Sistem pakar	4 / 5 (80%)	5 / 5 (100%)
Fuzzy inference	2 / 4 (50%)	4 / 4 (100%)
Simple additive weighting	3 / 4 (75%)	4 / 4 (100%)
Sistem pendukung keputusan	5 / 8 (62.5%)	8 / 8 (100%)
Pakan ternak	1 / 3 (33.3%)	3 / 3 (100%)
Metode klustering	1 / 5 (20%)	5 / 5 (100%)

Metode klasifikasi	0 / 8 (0%)	8 / 8 (100%)
Decision tree	0 / 3 (0%)	3 / 3 (100%)

7.4 Analisis

7.4.1 Analisis Hasil Pengujian Unit

Untuk pengujian *basis path* pada fungsi tambah koleksi didapatkan 2 kasus uji sudah menunjukkan hasil yang diharapkan. Pada fungsi ambil hasil pencarian menunjukkan 4 kasus uji sudah menunjukkan hasil yang diharapkan. Pada fungsi hapus koleksi didapatkan 2 kasus uji sudah menunjukkan hasil yang diharapkan.

Dari pengujian *basis path* didapatkan *cyclomatic complexity* atau V(G) terbesar dan terkecil yaitu 4 dan 2. Dari 3 buah pengujian *basis path* didapatkan rata-rata nilai *cyclomatic complexity* sebesar 2.66. Dengan nilai rata-rata V(G) tersebut diketahui bahwa sistem pencarian koleksi masih tergolong tidak terlalu rumit dalam pemeliharaan kode program.

7.4.2 Analisis Hasil Pengujian Validasi

Dari pengujian *black box* didapatkan hasil 100% dari 9 kasus uji. Hasil analisis dari pengujian validasi sistem pencarian koleksi dijelaskan pada Tabel 7.17.

Tabel 7.18 Hasil Pengujian Validasi

Kebutuhan	Kasus Uji	Hasil yang Didapatkan	Status
RBF-F-001	VAR-RBF-001	Sistem akan menampilkan informasi hasil pencarian koleksi.	Valid
RBF-F-001	VAR-RBF-002	Sistem akan mengirim notifikasi bahwa kolom *search field* kosong dan aktor tidak akan diarahkan ke halaman hasil pencarian.	Valid
RBF-F-002	VAR-RBF-003	Sistem akan menampilkan notifikasi bahwa koleksi baru berhasil ditambahkan dan sistem akan	Valid

		memperbarui halaman daftar koleksi.	
RBF-F-002	VAR-RBF-004	Sistem akan menampilkan notifikasi bahwa terdapat *field* kosong dan sistem tidak akan melakukan apapun.	Valid
RBF-F-003	VAR-RBF-005	Sistem akan menampilkan halaman daftar koleksi.	Valid
RBF-F-004	VAR-RBF-006	Sistem akan menampilkan notifikasi bahwa koleksi baru berhasil diperbarui dan sistem akan memperbarui halaman daftar koleksi.	Valid
RBF-F-004	VAR-RBF-007	Sistem akan menampilkan notifikasi bahwa terdapat *field* kosong dan sistem tidak akan melakukan apapun.	Valid
RBF-F-005	VAR-RBF-008	Sistem akan menampilkan notifikasi bahwa data koleksi berhasil dihapus dan sistem akan memperbarui halaman daftar koleksi.	Valid

RBF-F-005	VAR-RBF-009	Sistem akan menutup halaman *popover* dan proses hapus data koleksi dibatalkan.	Valid

7.4.3 Analisis Hasil Pengujian Perbandingan Pencarian Sintaksis dan Semantik

Dari hasil pengujian perbandingan pencarian sintaksis dan semantik bagian pertama didapatkan bahwa dari seluruh kasus uji didapatkan hasil yang sesuai dengan yang diharapkan. Namun pengujian ini tidak dapat menunjukkan kekuatan semantik pada hasil pencarian karena hanya melakukan pencarian secara sintaksis. Sedangkan pada pengujian bagian kedua berdasarkan Tabel 7.16 menunjukkan bahwa seluruh data dari tiap subjek dapat diperoleh dengan menggunakan teknik semantik.

BAB 8 PENUTUP

8.1 Kesimpulan

Dari hasil penelitian yang telah dilakukan, maka dapat diambil kesimpulan sebagai berikut:

1. Sistem dapat dianalisis untuk mengidentifikasi aktor yang terlibat dan mendapatkan kebutuhan fungsional. Dari hasil analisis kebutuhan kemudian dirancang ke dalam *use case diagram*, *use case scenario*, *sequence diagram*, *class diagram*, perancangan data, dan perancangan antarmuka.
2. Sistem dapat diimplementasikan berdasarkan perancangan yang telah dilakukan dengan menggunakan pendekatan berorientasi objek (*object-oriented*). Ontologi yang sudah dirancang sebelumnya diimplementasikan didalam Protégé, kemudian di-*export* dan dimasukkan sebagai *dataset* pada Apache Jena Fuseki.
3. Sistem dapat diuji dengan menggunakan metode *black-box* dan *white-box*. Pada pengujian *basis path* didapatkan hasil 100% *valid* dari 8 kasus uji. Pada pengujian validasi dengan metode *black-box* didapatkan hasil 100% *valid* dari 9 kasus uji. Pada pengujian perbandingan pencarian sintaksis dan semantik bagian pertama didapatkan bahwa dari seluruh kasus uji didapatkan hasil yang sesuai dengan yang diharapkan dan pada pengujian bagian kedua menunjukkan bahwa seluruh data dari tiap subjek dapat diperoleh dengan menggunakan teknik semantik.

8.2 Saran

Berdasarkan penelitian yang telah dilakukan, saran yang diberikan untuk penelitian selanjutnya adalah sebagai berikut:

1. Memperkaya *domain* ontologi untuk memberikan makna semantik yang lebih luas pada *domain* koleksi.
2. Menambahkan kecerdasan buatan pada sistem untuk rekomendasi kata kunci dan menghasilkan hasil pencarian yang lebih relevan.

DAFTAR PUSTAKA

Antoniou, G., dan Frank van Harmelen. 2003. *A Semantic Web Primer.* London: The MIT Press.

Arwan, Achmad, Bayu Priyambadha, Riyanarto Sarno, Mohamad Sidiq, dan Heri Kristianto. 2013. "Ontology and Semantic Matching for Diabetic Food Recommendations." (Information Technology and Electrical Engineering (ICITEE)). Diakses Agustus 13, 2017.

Awaludin, M. 2010. *Sistem Navigasi dan Pencarian Berbasis Konteks pada Konten Elearning Menggunakan Teknologi Web Semantik.*

Fensel, D., Frank van Harmelen, I Horrocks, D. L. McGuinness, dan P. F. Patel-Schneider. 2001. "OIL: an ontology infrastructure for the Semantic Web." *IEEE Intelligent System.* Diakses Februari 21, 2017. http://ieeexplore.ieee.org/document/920598/.

Fuseki. 2017. *Apache Jena Fuseki Documentation.* Diakses Februari 20, 2017. http://jena.apache.org/documentation/fuseki2.

Ghaleb, F. F. M., S. S. Daoud, A. M. Hasna, J. M. Jaam, dan H. F. El-Sofany. 2006. "A Web-Based E-Learning System Using Semantic Web Framework." *Journal of Computer Science* 2. Diakses Februari 21, 2017. https://www.researchgate.net/publication/26445641_A_Web-Based_E-Learning_System_Using_Semantic_Web_Framework.

Goldschmidt, David E., dan Mukkai Krishnamoorthy. 2005. "Architecting a Search Engine for the Semantic Web." *American Association for Artificial Intelligence.*

Ibrahim, N. 2007. "Pengembangan Aplikasi Semantic Web Untuk Membangun Web yang Lebih Cerdas." *Jurnal Informatika* 3. Diakses Februari 2017, 22. http://majour.maranatha.edu.

Lanin, V., dan L. Lyadova. 2007. "Intelligent Search and Automatic Document Classification and Cataloging Based on Ontology Approach." *International Journal Information Theories & Applications.* Diakses Februari 21, 2017. https://www.researchgate.net/publication/229047614_Intelligent_Search_and_Automatic_Document_Classification_and_Cataloging_Based_on_Ontology_Approach.

Maedche, A., dan S. Staab. 2005. "Ontology learning for the Semantic Web." *IEEE Intelligent Systems* 16 (2). Diakses April 11, 2017. http://ieeexplore.ieee.org/document/920602/.

Nilsson, M., A. Powell, P. Johnston, dan A. Naeve. 2008. "Expressing Dublin Core metadata using the Resource Description Framework (RDF)." *Dublin Core Metadata Initiative.* Diakses Februari 21, 2017. http://dublincore.org.

Pressman, Roger S. 2001. *Software Engineering: A Practitioner's Approach.* 5th. New York: McGraw-Hill Higher Education.

W3C. 2017. *Inference.* Diakses Agustus 21, 2017. http://www.w3.org/standards/semanticweb/inference.